This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2022.3187149

S-Glint: Secure Federated Graph Learning with Traffic Throttling and Flow Scheduling

Tao Liu, Peng Li, *Senior Member, IEEE,* Yu Gu, *Senior Member, IEEE,* Zhou Su, *Senior Member, IEEE,*

***Abstract*—Federated learning has been proposed as a promising distributed machine learning paradigm with strong privacy protection on training data. In this paper, we study federated graph learning (FGL) under a cross-silo setting where several servers are connected by a wide-area network and cooperate in training a graph convolutional network (GCN) model. We find that communication becomes the main system bottleneck because of frequent information exchanges among federated severs and limited network bandwidth. To tackle this challenge, we design S-Glint, a secure federated graph learning system, containing a homomorphic encryption-based protocol with pre-aggregation and batching strategies. The network traffic throttling and flows scheduling are also proposed to accelerate the learning process. To evaluate the effectiveness of S-Glint, we conduct experiments using trace-driven simulations. The results show that S-Glint can significantly outperform existing federated learning solutions.**

***Index Terms*—Federated learning, graph convolutional networks, homomorphic encryption, flow scheduling.**

I. Introduction

FEDERATED learning (FL) has shown great promise in enabling collaborative machine learning among distributed devices while preserving their data privacy [1]. When federated learning meets 6G networks, they can benefit each other while new challenges also emerge, which motivate us to study the communication efficiency of federated learning in this paper.

Existing federated learning mainly focuses on the convolution neural network (CNN) models that show superior learning accuracy in recognizing images and voices. However, many applications generate graph data (e.g., social graphs and protein structures) consisting of vertices and edges, which cannot be efficiently handled by CNN. Graph convolutional network (GCN) model [2] has been proposed to deal with graph learning by filtering the features of neighboring vertices.

In this paper, we study federated graph learning (FGL) under a cross-silo setting, including a set of FL servers maintained by different institutions (e.g., banks, hospitals, and universities). These servers may reside in public or private clouds, and they are connected by a wide-area network (WAN). Each server maintains a graph with edge connections to the others, and they cooperate in training a GCN model.

The FGL studied in this paper faces two main challenges. First, GCN training involves sharing features of neighboring graph nodes between FL servers, which leads to privacy leakage. Our previous work [3], [4] protects privacy by eliminating node feature sharing in the first layer of GCN. However, abandoning these node features may reduce learning accuracy. Second, there are massive devices in 6G, and existing works have shown that communication becomes the main bottleneck of distributed machine learning, which seriously affects the efficiency of learning tasks [1]. The communication challenge faced by FGL is more severe than traditional CNN-oriented federated learning. In addition to trainable parameter synchronization, servers in FGL need to exchange vertex embeddings in every graph convolutional layer during each training round due to its unique characteristics. Moreover, FGL relies on a wide-area network, which is shared by many applications and bandwidth allocated to FGL is limited and dynamic.

Although various methods [5], [6] have been proposed to optimize communication, they differ from our problem studied in this paper. The most significant difference is that they usually only study the trade-off between learning speed and energy efficiency [5], [6]. Privacy is not a concern. And the previous cross-silo related work [7] does not exploit GCN's characteristics about frequency embedding exchange. We also notice that some distributed systems [8], [9] conquer the communication issue by jointing data movement and task allocation schemes. In contrast, due to privacy protection, FGL does not allow such graph data movement. There also exist some general flow scheduling approaches like conventional shortest-flow-first [10], but they bring limited improvements. Some other complex scheduling approaches claim a better performance [11]–[13]. However, it should be quite challenge to fully exploit the unique characteristics of FGL.

We propose S-Glint, a federated graph learning system with enhanced security insurance and a series of novel designs to address the communication challenge. In order to protect vertex embeddings, S-Glint adopts a homomorphic encryption (HE) based protocol that only transmits encrypted data. We adopt pre-aggregation and batching strategies with multiple HE servers to accelerate the encryption process. S-Glint further tackles the communication challenge with two novel designs. First, it reduces the network traffic by eliminating the

T. Liu and P. Li are with School of Computer Science and Engineering, The University of Aizu, Japan. Email: {d8212107, pengli}@u-aizu.ac.jp.

Y. Gu is with S^2AC Lab, School of Computer and Information, Hefei University of Technology, Key Laboratory of Knowledge Engineering with Big Data, Hefei University of Technology, Hefei, China. Email: yugu.bruce@ieee.org.

Z. Su is with Xi'an Jiaotong University. Email: zhousu@ieee.org.

This research was supported by JSPS KAKENHI (No. 21H03424), the Basic Business Expenses of Central Public Welfare Scientific Research Institutes of the Chinese Academy of Medical Sciences under No.2020-JKCS-002, the Key Research and Development Plan of Anhui Province under Grant No.202004b11020018

0000–0000/00$00.00 © 2021 IEEE

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 11,2022 at 08:19:27 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2022.3187149

transmission of unimportant embeddings, also referred to as traffic throttling. Specifically, S-Glint lets each node quickly evaluates its neighbors' contributions based on marginal loss and only collects data from a subset of essential neighbors. The second novel design is a priority-based dynamic flow scheduling strategy. S-Glint monitors the training speed of FL servers and dynamically assigns different priority levels to network flows. Besides, traffic throttling and flow scheduling are jointly considered for further time savings. The main contributions of this paper are summarized as follows:

1) We propose a secure federated graph learning system called S-Glint to enable collaborative training of GCN models among distributed servers without leakage of their local graph data. We have identified that communication is the main bottleneck of the decentralized, federated graph learning system.
2) We analyze the graph training characteristics and propose a homomorphic encryption-based embedding sharing strategy for safety and efficiently embedding interaction. We design the traffic throttling module to eliminate unimportant transmission. Besides, the flow scheduling problem is formulated and analyzed. We further combine traffic throttling and heuristic flow scheduling for joint optimization.
3) We simulate S-Glint based on trace data and evaluate its performance using a 20-server setting. Extensive experimental results show that S-Glint can significantly outperform existing works.

The rest of this paper is organized as follows. In Section II, we first give some background and motivation. Then we present the system design in Section III. The experimental results are presented in section IV, followed by some discussions in Section V. The related work is in Section VI. We conclude our work in Section VII.

II. Background and Motivation

A. Federated Learning

Federated learning (FL) has been proposed to enable collaborative training among multiple servers without leaking any raw data. Its basic idea is to let servers share model parameters instead of training data. A typical parameter synchronization scheme widely adopted by federated learning is the Parameter Server (PS) [14]. Specifically, a federated learning system consists of a parameter server and a set of computing servers. The training process contains multiple rounds. Every server uses its local dataset to train a model in each round. After local training, servers send their local models to the parameter server, and the parameter server then creates a new global model and distributes it for the next-round training. Despite the simplicity, PS-based federated learning suffers from the global synchronization bottleneck at the parameter server.

B. Graph Convolutional Network

Given a graph where each vertex is associated with a feature vector, a graph convolutional network (GCN) aims to transfer vertex feature vectors into compressed ones, also called embeddings, by exploiting graph structure and vertex features. GCN stacks multiple layers, and each layer has the same structure as the original graph. The graph convolution operation is defined as aggregating node embeddings of neighboring nodes. We use A to denotes the graph's adjacent matrix and H^l to denotes the matrix of node embeddings in the l-th layer. The propagation rule of the GCN can be formally expressed by

$$z^{l+1} = \overline{A} H^l W^l, \qquad H^{l+1} = \sigma(z^{l+1}), \tag{1}$$

where $\overline{A} = D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}$, W^l denotes trainable feature weights, D is the degree matrix and σ is the activation function.

Given a set V of nodes with labels, the GCN training goal is to minimize the loss function:

$$\mathcal{L} = \frac{1}{|\mathcal{V}|} \sum_{j \in \mathcal{V}} \mathcal{F}(y_j, \hat{y}_j), \tag{2}$$

where y_j and $\hat{y}_j$ denote the true label and the predicted one, respectively. The loss function $\mathcal{F}$ usually uses the cross-entropy.

C. Homomorphic encryption

Homomorphic encryption (HE) is designed for carrying out function computation on ciphertexts while preserving the functional characteristics of the plaintext [15]. A typical homomorphic encryption scheme can be expressed as $HE = (KeyGen, Enc, Dec, Eval)$, where the function $KeyGen$ generates a public key pk and a private key sk, Enc and Dec are the encryption function and the decryption function, respectively. The computation operations supported by homomorphic encryption are denoted by $Eval$. The characteristic of homomorphic encryption can be described as $D_{sk}(Eval(E_{pk}(a), E_{pk}(b))) = Eval(a, b)$, which indicates that by decrypting the results of operations on ciphertext, we can obtain the same results with those of original operations on plaintext. Due to this unique characteristic, homomorphic encryption is usually used for outsourcing computation to the third-parties.

D. Communication Bottleneck in FGL

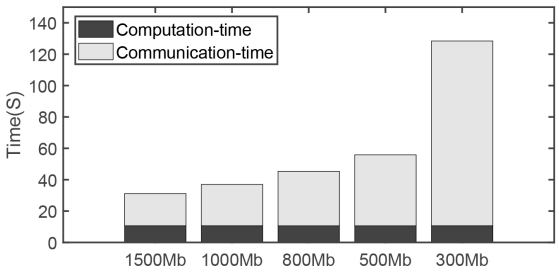
It has been well recognized that communication is the main bottleneck of distributed machine learning, especially in the WAN environment [16]. To better understand how communication affects FGL, we have conducted some preliminary empirical studies on a 3-server cluster, where each server is equipped with Inter i7-10700 CPU, 16GB memory, and Geforce RTX 2080 GPU. A switch connects three servers, and the network bandwidths of all links are set to 1500Mbps. We divide the widely used Reddit graph dataset (232965 vertices with 602 features for each vertex) into three parts stored by these servers, respectively. Each server creates a two-layer GCN based on its local graph (including connected vertexes held by others) and exchanges trainable parameters after each training round. The amount of data exchange averaged over training rounds are shown in Figure 1. The size of trainable

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 11,2022 at 08:19:27 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2022.3187149

		Server 1	Server 2	Server 3
Server 1	Embedding	0	443 MB	440 MB
	Local model	0	0.25 MB	0.25 MB
Server 2	Embedding	416 MB	0	431 MB
	Local model	0.25 MB	0	0.25 MB
Server 3	Embedding	408 MB	427 MB	0
	Local model	0.25 MB	0.25 MB	0

(a) The amount of transmitted data.



(b) Time cost with various network bandwidths.

Fig. 1: Example of cooperate training among three servers.

parameters is about only 0.25Mb, while exchanged embedding size is over 400Mb.

We then change the network bandwidth and study how it affects training time. As shown in Fig. 1, we can see that the communication time increases dramatically as bandwidth decreases, while the computation time is almost the same. Other graph data in practice may involve even billions of nodes and edges [17], which makes the FGL more challenging.

III. System Design

A. Overview

We consider a typical federated setting consisting of a set C of distributed servers connected by a WAN with limited bandwidth. Each FL server $c_i \in C$ holds a local training dataset expressed as a graph $\mathcal{G}_i = (V_i, E_i)$, where V_i and E_i denote the vertex set and edge set, respectively. Every vertex $v \in V_i$ is associated with a feature vector x_v that cannot be exposed to other servers. Note that there exist some graph edges across servers. Each server is aware of the existence of connected vertices at other servers but cannot access their vertex feature vectors. For example, suppose there are some hospitals and a medical administration center. The global graph records, for a certain period, the city's patients (nodes), their information (attributes), and interactions (links). Specifically, we have two kinds of links, the first one is explicitly stored in each hospital, and the second one is the cross-hospital links that may exist but are not stored in any hospital. The medical administration center may have the goal that the system obtains a globally powerful graph mining model while not sharing data.

Based on the above scenario, we propose a secure federated graph learning system, S-Glint. S-Glint adopts a decentralized design to coordinate model parameter sharing and embedding sharing. At the beginning of each training round, each FL server collects model parameters from others and then averages them to get the initial weights used in the current-round training, eliminating the global barrier of the centralized parameter server in traditional federated learning. The schematic diagram of S-Glint is shown in Fig. 2. There are three critical designs in S-Glint. The first is neighbor selection. For each FL server, its neighbors may have various contributions; S-Glint aims to evaluate each FL server's neighbors and select essential neighbors. The second is data encryption. S-Glint adopts the HE mechanism with pre-aggregate and batching operations to efficiently encrypt transmitted data, thus protecting users' privacy. In the conference version [3], the features of nodes are multiplied by the weight at the second GNN layer. The decreased dimension thus protects privacy. We claim that this method mainly contains two weaknesses. First, although the above method is workable in most situations, it has a high risk of being exposed when the feature matrix is sparse since it lacks rigorous safety theory proof. Second, although the conference version considers the neighbors' information from the second layer, ignoring the first GNN layer harms the accuracy. In our experiments, the dimension decrease method causes a little accuracy decrease. The gap needs to be made up. The third key design of S-Glint is flow scheduling. S-Glint analyzes the flow scheduling problem in our scenario and proposes a heuristics scheduling method with co-design to neighbor selection design. The details of the three designs are described in the following subsections.

B. Secure embedding sharing

For each FL server c_i, the set of nodes with edge connections to graphs held by other servers is denoted by V_i^n, whose embeddings are encrypted as $\{En_{pk}[h(v)], \forall v \in V_i^n\}$. These encrypted embeddings are sent to a HE server who aggregates them according to requirements of FL servers. For example, FL server $c4$ requests embeddings of nodes v_3 from server $c1$ and v_5 from server $c2$, the HE server aggregates their encrypted embeddings as $En_{pk}[h(v_3)]+En_{pk}[h(v_5)]$ and sends the result to the $c4$.

1) Pre-aggregate: We find that the above process has high communication overhead due to the transmission of a large amount of encrypted node embeddings from FL servers to the HE server. By carefully examining this process, we find that the communication overhead can be reduced if FL servers can aggregate some embeddings before encryption. For example, if two shared node sets of different parties have repeated nodes, the repeated nodes can be aggregated first to avoid being encrypted and transmitted separately.

Motivated by the above example, we express embedding requests as a matrix and derive its basis. Then, each FL server encrypts its embeddings according to this basis and sends them to the HE server. Any linear combination of these encrypted embeddings at the HE server can be computed and shared with other FL servers.

2) Batching: We find that the ciphertexts generated by HE are significantly larger than plaintexts. Large ciphertexts incur high communication overhead. That is because ciphertexts have roughly the same number of bits with the key size rather

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 11,2022 at 08:19:27 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2022.3187149

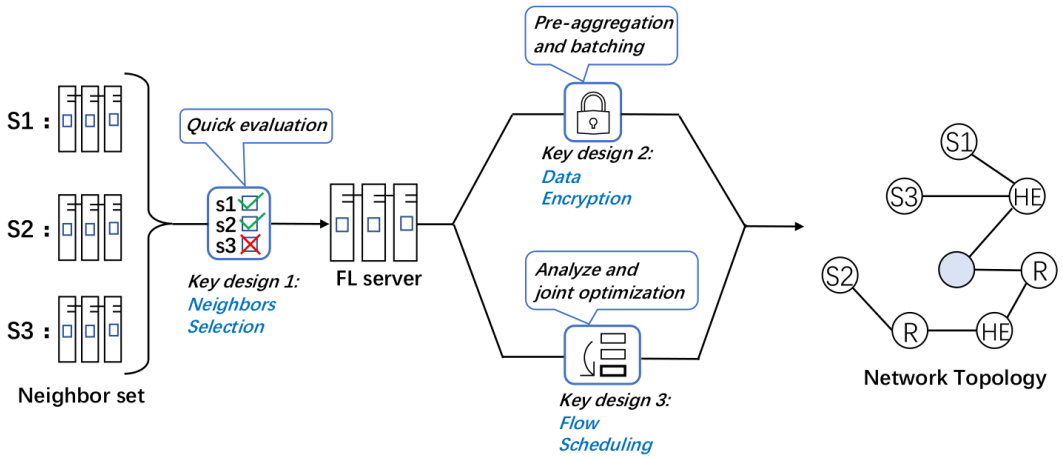


Fig. 2: The architecture of S-Glint.

than plaintext size. For example, graph embedding values are typically 32-bit, but the key size of Paillier, a popular HE lib, is 2048. To reduce ciphertext size, we adopt the batching technique [18] that concatenates several embedding values to encrypt them together so that they share the same plaintext while preserving the additive property.

3) Multiple HE servers: The straw-man design has a single HE server, which would be the communication bottleneck since all shared embeddings, in the encrypted form, need to go through it. We deploy multiple HE servers to distribute the communication burden and eliminate this bottleneck. Specifically, we set one HE server and several FL servers become a group. The HE server is responsible for the incoming data flow for all FL servers in the group (including the data interaction among the FL servers in the same group). The whole system then contains multiple groups and still works in a decentralized way.

C. Traffic Throttling

In S-Glint, each server collects feature embeddings from its neighbors. These neighbors have different contributions to the training process. We divide the commonly used Cora dataset (containing 2708 nodes and 10556 edges) into four parts stored by four servers. Server 1 and server 2 can train a model with 50% accuracy, while server 1 and server 3 achieve 60%. The basic idea of traffic throttling is to evaluate neighbors' contributions and eliminate unimportant data transmissions. Traffic throttling mainly contains two phases: contribution evaluation and neighbor selection, whose pseudo-codes are shown in Algorithm 1.

1) Contribution Evaluation: We follow the idea in [19] to quantify the contribution of shared embeddings. Note that the network flows associated with model sharing can be throttled in a similar way. Specifically, in each epoch r, server c_i first receives embeddings from all neighbors in set N_i and conducts training to generate a local model $M^r_{N_i}$. Then, for each neighbor $c_j \in N_i$, sever c_i also trains a model $M^r_{(N_i-j)}$,

Algorithm 1 Experience-based Traffic Throttling.

INPUT: C is the set of all servers, N_i contains the neighbors of server c_i, Epo is the total training epochs, and The previous k epochs are utilized for evaluation.
OUTPUT: Selected neighbors of every c_i.
\# $The\ contribution\ evaluation\ phase$:
for server $c_i \in C$ in parallel **do**
  for $r = 1, 2, \ldots, k$ **do**
    c_i calculates neighbors' embedding contributions scores with (3).
    c_i calculates neighbors' weight contributions scores with (3).
  end for
end for
\# $The\ neighbors\ selection\ phase$:
for each server $c_i \in C$ in parallel **do**
  Selects the neighbors according to the joint optimization with the threshold ψ.
  for $r = k+1, k+2, \ldots, Epo$ **do**
    Collects information from the selected neighbors.
    Estimates the unselected neighbors' embedding and model weight using (4) and (5), respectively.
    Conduct the local GCN training.
  end for
end for

where $N_i - j$ means the neighbors set excluding the server c_j. The embedding contributions of c_j thus can be calculated as:

$$S_{j,i} = \frac{\sum_{r=1}^{k}[L(M^r_{N_i}) - L(M^r_{N_i-j})]}{\sum_{j\in N(i)}\sum_{r=1}^{k}[L(M^r_{N_i}) - L(M^r_{N_i-j})]}, \quad (3)$$

where $L(M_{N_i})$ denotes the training loss of model M_{N_i}. Both models $M^r_{N_i-j}$ and $M^r_{N_i}$ are trained based on the $M^{r-1}_{N_i}$ in the previous epoch. Note that k is the number of epochs used for contribution evaluation.

The contribution evaluation process can be completed quickly with several reasons and strategies. Firstly, we claim that a few epochs (threshold k) are enough to evaluate the

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 11,2022 at 08:19:27 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2022.3187149

contributions, and it will be verified in Section IV. Secondly, the formulation (3) does not involve the process of complete backpropagation and verification, which will save time. Thirdly, compared to CNN, the GCN model is lightweight enough that the overhead in computation time is limited even in the case of multiple executions. Finally and most importantly, we propose caching strategy to accelerate the process. There are many repeated nodes in the evaluation, and we can save the repeated nodes in the GPU to avoid reloading and reduce communication and computation overhead.

2) Neighbor Selection: The neighbor selection phase selects neighbors with high contributions and estimates the training input accordingly based on contribution evaluation results. In our conference version [3], we let each FL server sorts its neighbors according to their contributions scores and selects those with larger contributions until their total scores exceed the threshold ψ. However, this design may demand data transmission from a slow neighbor, while two faster neighbors may replace the contribution of this slow neighbor and save time. Thus we only expect the sum of the contribution to exceed the given threshold here. The specific neighbor selection needs to jointly consider the communication conditions and arriving speed of incoming flows. We will talk about the details in the following section.

As for ψ, our experimental results in Section IV show that an appropriate threshold can speed up the training process with minor accuracy reduction. After neighbor selection, we estimate the embedding used for training input according to the collected ones from selected neighbors. For each server c_i with α_i flows, it extracts the embedding information contained in these flows and updates the local adjacent matrix $\overline{A}_{uv}$ as follows.

$$\overline{A}_{uv}^{(l-1)} = \begin{cases} \frac{|N(u)|}{|N'(u)|}\overline{A}_{uv}, & \text{if } v \in N(u), \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

where $N(u)$ denotes the set of neighbors of vertex u in the original graph, and $N'(u)$ denotes the set of neighbors obtained with α_i flows. Note that both $N(u)$ and $N'(u)$ contain internal neighbors and external neighbors maintained by other servers. For weight sharing stages, we estimate the initial weights with β_i flows, whose senders are included in set $\mathbb{C}_i$:

$$\omega_i = \sum_{c_j \in \mathbb{C}_i} \frac{|V_j| w_j}{\sum_{c_j \in \mathbb{C}_i} |V_j|}, \tag{5}$$

where V_j is the set of labeled vertices held by server c_j.

D. Flow Scheduling

It has been well recognized that flow scheduling is significant for communication efficiency [5]. The intuitive idea is to formulate the scheduling process as an optimization problem. Thus we first formulate and analyze the problem.

1) Formulation: Suppose the data amount of flow f_{ij}^s is D_{ij}^s, which means the server c_i needs transmit D_{ij}^s data to the server c_j at training stage s. We suppose the server's index that takes the longest time to complete the total S training stage is 0. We let T_j^s denotes the completion time of the server c_j in the stage s. To minimize the total training time T, i.e., the completion time of the final stage, we formulate the flow scheduling problem as follows:

$$min T_0^S; \tag{6}$$

$$T_j^s = \max_t\{T_{ij}^s\} + t_{j,cpt}^s, \forall j, s; \tag{7}$$

$$T_{ij}^s = \max_t\{t \cdot x_{ij}^s[t]\}, \forall f_{ij}^s; \tag{8}$$

$$x_{ij}^s[t] \geq \frac{y_{ij}^s[t]}{D_{ij}^s}, \forall i, j, s, t; \tag{9}$$

$$\sum_t y_{ij}^s[t] \geq D_{ij}^s, \forall f_{ij}^s; \tag{10}$$

$$\sum_{f_{ij}^s \in F(p)} y_{ij}^s[t] \leq B_p \forall t, p; \tag{11}$$

$$x_{ij}^s[t]\hat{T}_{ij}^s \leq t \cdot x_{ij}^s[t], \forall f_{ij}^s, t; \tag{12}$$

$$T_i^{s-1} \leq \hat{T}_{ij}^s, \forall f_{ij}^s, t; \tag{13}$$

Our objective is to minimize the completion time of the final stage, which is expressed in constraint (6). For each stage s, the completion time of the server c_j is calculated by (7), where T_{ij}^s denotes the completion time of flow f_{ij}^s and $t_{j,cpt}^s$ means the computation time of the server c_j for the stage s. We define a binary variable $x_{ij}^s[t]$ to denotes whether we transmit the flow f_{ij}^s in the time slot t. Then the flow completion time T_{ij}^s can be expressed by (8). Another integer variable $y_{ij}^s[t]$ is defined to denote the amount of transmitted data of flow f_{ij}^s in the time slot t, whose relationship with $x_{ij}^s[t]$ is shown in (9). If f_{ij}^t is transmitted in time slot t, i.e., $y_{ij}^s[t] > 0$, the binary variable $x_{ij}^s[t] = 1$. Otherwise, we have $x_{ij}^s[t] = 0$ due to the minimization objective. Constraint (10) presents that the total amount of transmitted data should be no less than the flow size. Due to the bandwidth constraint of each network links, we have constraint (11), where $F(p)$ denotes the set of flows going throughput the network link p. We let $\hat{T}_{ij}^s$ denotes the start time of flow f_{ij}^s and it should be no less than the completion time T_i^{s-1} of the previous stage, which is represented by constraints (12) and (13).

The above formulation is hard to be solved directly. It is a mixed-integer nonlinear programming problem, which is generally NP-hard. Although it is easy to transfer it into a linear form by applying some internalization techniques, it is still challenging to solve this large-scale optimization problem since federated graph learning contains many stages. Besides, the network bandwidth B_p could be dynamic because many applications share the network.

2) Insights: Although it is challenging to find the optimal scheduling solution, there are some heuristic insights when revisiting the federated graph learning details. First, the flows belonging to the earlier stages should be transmitted preferred because they can enable the received server to finish the current stage computation faster. Second, we need to consider the computation time of the servers after receiving flows. If a server needs a longer time for training, its incoming flows should have a higher priority to avoid being the bottleneck.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 11,2022 at 08:19:27 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2022.3187149

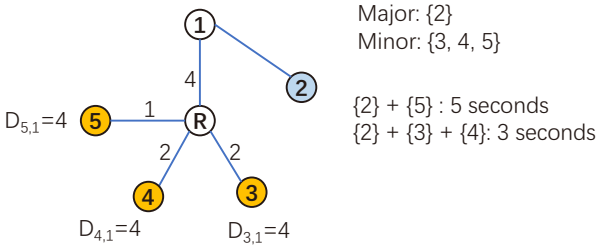


Fig. 3: An example of joint optimization.

Algorithm 2 Joint Optimization.

INPUT: There are P priority levels. Each server c_i divides its output flows into major flow set $\{F_{i_1}\}$ and minor flow set $\{F_{i_2}\}$. c_{j_1} is the major neighbor set of c_i, c_{j_2} is the minor neighbor set of c_i. C is the FL server set.
OUTPUT: The final transmission time cost.
for server $c_i \in C$ in parallel **do**
  # *The first period*
  When c_i starts its $r-th$ training epoch, c_i collects stage set St_i, time set T_i and records their size Len_i.
  c_i distributes priorities of flows in $\{F_{i_1}\}$ according to (14) and (15).
  c_i distributes lowest priorities of flows in $\{F_{i_2}\}$.
  if a server c_i receive all major neighbors' data **then**
    # *The second period*
    for server $c_j \in c_{j_2}$ **do**
      if c_j transmit data to c_i before **then**
        c_j will be selected
      end if
      select servers in c_{j_2} according to estimated arrival time.
    end for
    Turn the selected neighbor in minor to major and distributed priorities according to (14) and (15).
  end if
end for

Since the neighbor selection decision in the previous traffic throttling module has multiple possibilities, which will seriously affect the flow scheduling module, S-Glint adapts a joint optimization scheme to combine the neighbor selection and flow scheduling. The details are described as follows.

3) Joint Optimization: Our conference version [3] only considers the above two insights into a dynamic flow scheduling strategy while ignoring the selection of neighbors. Here we propose a joint heuristic scheme. The process is shown in Algorithm 2. For a FL server c_i, we classify its neighbors as two sets, $\{c_{j_1}\}, j_1 \in N_i$ and $\{c_{j_2}\}, j_2 \in N_i$. $\{c_{j_1}\}$ contains the major neighbors that their embedding information are indispensable. c_{j_2} contains the minor neighbors that part of them may be selected to match the contribution threshold.

Fig. 3 gives an example of neighbors selection. Suppose server 1 has the major neighbor 2 and minor neighbor set $\{3, 4, 5\}$. Suppose the contribution of neighbor 5 is larger than neighbor 3 and neighbor 4, and the server has two choices to match the contribution threshold, $\{2\} + \{5\}$ or $\{2\} + \{3, 4\}$. Obviously, if we still select neighbors according to the contribution order, neighbor 5 will be selected, and it will take 5 seconds for minor neighbors. Otherwise, if we choose $\{3, 4\}$, it only takes 3 seconds.

Since the selection of neighbors is related to both their contributions and arrival time, we separate our joint optimization into two periods. Specifically, we let each FL server receives all neighbors' data for the first period, and a dynamic priority-based flow scheduling method has been adopted. The main idea of the scheduling is that all the major neighbors should prioritize over the minor neighbors, and we should arrange proper priorities among the major neighbors. We suppose the system provides P priority levels that range from 0 to $P-1$, where 0 means the highest priority level. In training epoch r, server c_i maintains a set of flows F_i to be transmitted to other servers, which can be divided as major flows set F_{i_1} and minor flows set F_{i_2} according to the major or minor neighbor set it belongs. It also collects the information of corresponding receivers' current training stage and previous training stage time cost of the flows in F_{i_1} and maintains them in set St_i and T_i, respectively. We let Len_i denotes the size of St_i. We first sort the flows in F_{i_1} according to their receivers' training stages. The flows are ordered from the smallest training stage value to the largest one, and the smaller one means a larger priority. For the flows whose receivers are in the same training stage, we describe them as a sub-flow set and further sort the flows in each sub-flow set according to the time cost information. The sorted F_{i_1} is denoted as F'_{i_1}. For a flow f_j in F'_{i_1}, we calculate its priority P_j as follows:

$$ind = \lceil Len_i / P \rceil, \tag{14}$$

$$P_j = \lceil j / ind \rceil. \tag{15}$$

Then, the flow f_j is put into the P_j priority level queue. Note that all flows belonging to F_{i_2} are also sorted and only put into the $P-1$ priority.

Once a server obtains all flows from its major neighbors, it starts the second period. The server needs to choose neighbors in the minor set in the second period. If a neighbor in the minor set has already transmitted some data to the server previously, it will be chosen, like neighbor servers $\{3, 4\}$ in the previous example, and the rest minor neighbors whose estimated arrival time is the shortest will also be selected until the sum of the contributions exceeds the threshold. If all the neighbors in the minor set have no data transmission before, all selections are based on estimated arrival time. The selected neighbors in the minor set then turn to the major neighbor, and their priority level will also be distributed from the above principle. The transmission from unselected neighbors will be eliminated.

The above joint optimization process considers dynamic training speed and time cost to avoid the slower party becoming the bottleneck. It also ensures that the major neighbors are selected, and the minor neighbors are selected according to their coming speed to accelerate the process.

IV. Experiments and Evaluation

A. Experimental Settings

We simulate S-Glint based on PyTorch, and a python graph learning package, named Deep graph Library (DGL) [20]. The

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 11,2022 at 08:19:27 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2022.3187149

TABLE I: Graph Datasets

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	10,556	1,433	7
PubMed	19,717	88,651	500	3
Coauthor physics	34,493	991,848	3,703	6
Reddit	232,965	114,848,857	602	41

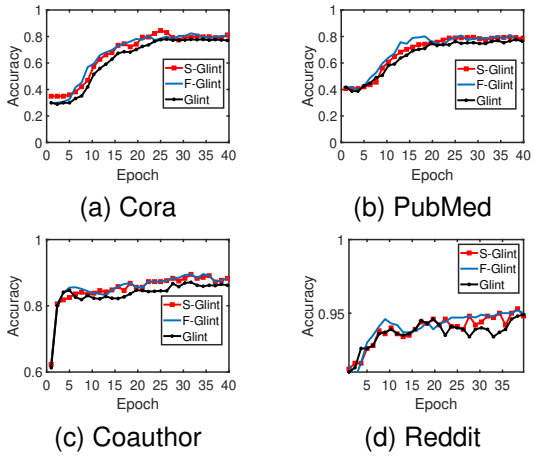


Fig. 4: Time-accuracy performance comparison.

hardware includes Inter i7-10700 CPU, 16GB memory, and Geforce RTX 2080 GPU. We choose four widely used graph datasets: Cora, PubMed, Coauthor, and Reddit, whose details are summarized in Table I. We extract a network topology from a real peer-to-peer network [21], which contains 20 FL servers, 10 HE servers, and 76 routers. Each HE server is responsible for the incoming data streams of two FL servers. The trace about data transmission among each part thus can be obtained through their real data interaction demand when we divide each dataset into 20 parties. Different links do not have identical capacity, and we set the bandwidth of links randomly from 5Mbps to 500Mbps. Note that network bandwidth may fluctuate over time, and network congestion can also happen. Thus we randomly set the bandwidth values of all links according to a Gaussian distribution, where the standard deviation is one-tenth of the mean. The threshold k, which represents the number of epochs for contribution evaluation, is set to 10. The threshold ψ, which denotes the sum of transmitted flows' contribution score, is set to 0.9. Each server randomly chooses a subgraph from a given dataset and trains a two-layer GCN model with the ADAM optimizer.

B. Results

We evaluate S-Glint in different dimensions with various baselines.

1) Accuracy Results: To evaluate the accuracy results of S-Glint, we choose two baselines. The first is F-Glint, where each FL server has no traffic throttling part. The second is our conference version, Glint [3]. Glint lets each FL server transmits the embedding information after the dimensional reduction in the second GCN layer to protect privacy and contains independent traffic throttling and flow scheduling strategies. The accuracy convergence of them over different

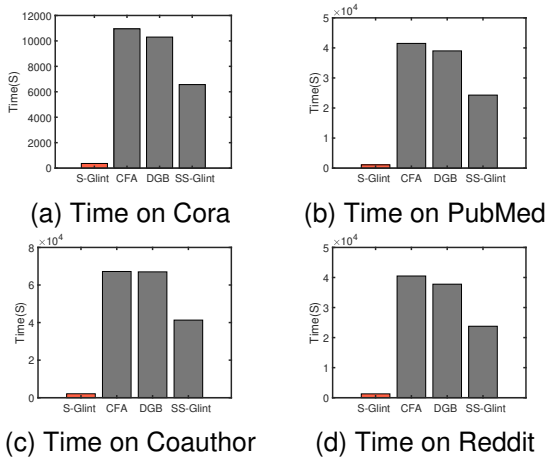


Fig. 5: Overall time cost comparison.

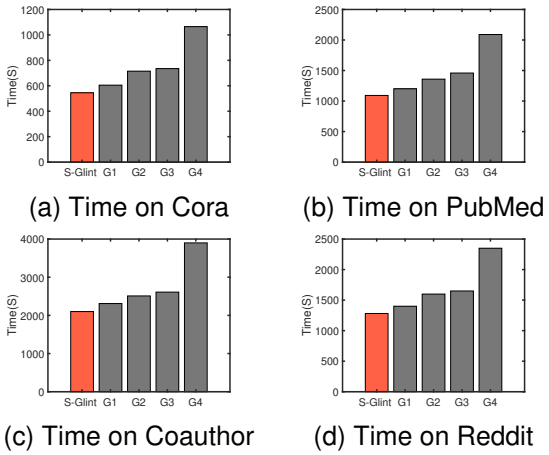
datasets is shown in Fig. 4. We can see that the accuracy of S-Glint is similar to the F-Glint, while Glint has a slight gap between S-Glint and F-Glint. For example, for the Cora dataset, S-Glint and F-Glint both have an accuracy of about 81%, while Glint is bout 80%. The reason is that Glint still ignores some information interaction since it only transmits data in the second GCN layer. F-Glint guarantees accuracy while does not consider privacy issues. S-Glint achieves similar accuracy to F-Glint and further efficiently encrypts the data to protect the privacy and utilizes joint optimization to release the communication burden.

2) Overall time cost: We also compare S-Glint with three baselines to evaluate the overall time cost. The first is centralized federated average graph learning (CFA). We extend FdeAvg [14], which has been widely used by many federated learning schemes, to implement a centralized federated average graph learning (CFA) scheme by adapting the single priority-based flow scheduling strategy like in Glint for the embedding exchanging part. It has a secure embedding sharing strategy without pre-aggregation and batching. The second is decentralized gossip-based federated graph learning (DGB). Combo [22] uses a segment operation and a gossip protocol to deal with local models' aggregation under a decentralized-federated learning scenario. The basic idea is to let the models take random walks in the network topology and get updated when they arrive at a server, which is different from ours. We extend Combo to implement a decentralized gossip-based federated graph learning (DGB) baseline with Glint's same single flow scheduling strategy. It also has a secure embedding sharing strategy without pre-aggregation and batching. The third is simple S-Glint (SS-Glint) with joint optimization, while the secure embedding sharing strategy has no pre-aggregation and batching.

We measure the completion time of different systems after 40 training epochs when they all have converged and show the results in Fig. 5. We can see that S-Glint achieves about 30x-40x speedup than other CFA and DGB. The improvement comes from the efficient, secure embedding sharing strategy and joint optimization. The performance of DGB is a little better than CFA because its local model aggregation needs

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 11,2022 at 08:19:27 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2022.3187149



(a) Time on Cora (b) Time on PubMed

(c) Time on Coauthor (d) Time on Reddit

Fig. 6: The effectiveness of the joint optimization.

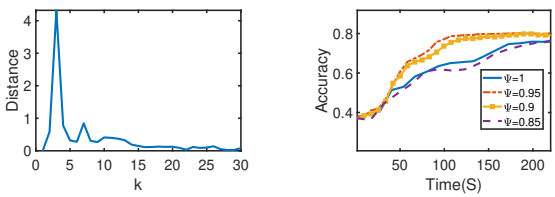
less communication time. However, the improvement is limited because local models are a small percentage of the whole network traffic, as shown in Fig. 1. The effectiveness of the encryption acceleration in our secure embedding sharing is just the gap between S-Glint and SS-Glint, which is about 10x-25x speedup. The traffic throttling and flow scheduling bring about 40% to 50% improvement when comparing CFA and DGB with SS-Glint.

3) The Effectiveness of the Joint Optimization: To further evaluate the joint optimization, we also choose several related baselines. Firstly, we modify S-Glint to separate the joint optimization part as independent traffic throttling and flow scheduling as in Glint, named G1. Secondly, we modify S-Glint to adopt traffic throttling and shortest-first scheduling, named G2. Thirdly, we modify S-Glint to adopt traffic throttling and fair-share scheduling, named G3. Finally, we modify S-Glint to abandon the traffic throttling part, named G4.

The results are shown in Fig. 6. We can find that the joint optimization brings about 10% improvement compared with the separate one. Even though we separate the joint optimization, there is also about 10% improvement compared with other scheduling methods. The specific traffic throttling strategy brings about 40% benefit when comparing S-Glint with G4.

C. The Influence of System Parameters

S-Glint's performance is also affected by system parameters, k represents the number of training rounds for contribution evaluation, and ψ denotes the threshold for neighbor selection in traffic throttling. We use the Cora dataset as an example to show the influence of parameters in Fig. 7. We conduct experiments to calculate the embedding contribution scores of server 1's neighbors. We first calculate the scores based on the complete training epochs. The scores of all neighbors thus can be formed as a vector. Then we compare the Euclidean distance between the above vector and the new vector calculated based on various k. When the value of k is larger than 10, their distance is stabilized and approaches 0. Similar phenomena occur in other datasets. Thus we set k to 10 in previous experiments.



(a) Various k (b) Various ψ

Fig. 7: The influence of system parameters.

For the threshold ψ, we set it as 1, 0.95, 0.9, 0.85, respectively. The choice of neighbors is based on the joint optimization until the sum score of neighbors exceeds the threshold. We can find that a smaller ψ may cause a large input error and further result in no coverage or slow down the training. We set ψ as 0.9, which has attractive performance on all datasets in practice.

V. Discussions

The Efficiency of the Caching Strategy. S-Glint introduces a caching mechanism in the neighbor contribution evaluation part to reuse the common nodes to accelerate the process. We claim that the caching strategy only brings very limited overhead. We use the tensor mask operation on GPU in the cache process to avoid complex index operation, thus having a minor increase in the computation overhead. In fact, the data loading from CPU to GPU takes the main time in the training of GNN [23], thus the limited computation overhead of our caching strategy is favorably offset by reducing the data loading communication.

The Overhead of the Encryption. Homoerotic encryption usually is a heavy operation with time-consuming. In S-Glint, we have two core strategies to reduce the encryption time. The first one is pre-aggregation. We do not encrypt every original node's features, instead, we express the request as a matrix and derive the basis. The encryption is according to the basis, thus can reduce the amount of encrypted data. In our experiments, The reduced ratio can achieve from about 10% (Cora dataset) to about 30% (PubMed dataset) when the dataset is divided into 20 parties.

The second strategy is batching. In HE, the size of ciphertexts is only related to the key rather than plaintext. The batching technique can concatenate plaintext and reduce the computation and communication overhead. We use the Cora dataset as an example to compare the S-Glint (with pre-aggregation and batching), SS-Glint (direct apply HE without pre-aggregation and batching), and P-Glint (plaintext with no encryption operation). Their overall time costs for the 40 epochs are about 560 seconds, 6280 seconds, and 230 seconds. We find that direct encryption brings a very high overhead on communication and computation, while the proposed strategies in S-Glint successfully reduce the overhead by order of magnitude.

The Synchronization in S-Glint. There are two kinds of transmitted data flows in the network. The first is the shared encrypted embedding, and the second is the encrypted weight. In a decentralized scheme, a naive method to synchronize the

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 11,2022 at 08:19:27 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2022.3187149

weight lets every participant transmit its model to all of the others, making the synchronization the basic bulk synchronous parallel (BSP). However, in S-Glint, we do not force this kind of fully-connected transmission. Instead, the traffic throttling strategy has been adopted to eliminate unnecessary transmission. Since we reduce the transmission, the communication overhead will be released. As the results are shown in the previous section, there is about 40% brought by the traffic throttling strategy. The benefit comes from eliminating some unnecessary and slow neighbors for each FL server.

The Overhead of Neighbors' Evaluation. In the contribution evaluation process, the FL server needs to perform forward propagation many times based on the marginal loss. We claim that this brings limited computation overhead due to the characteristic of GCN (small model) and our strategies, caching (quickly achieved by the tensor mask operation).

We compare the time cost of various components with a single training epoch of the Cora dataset, including forward-backward propagation (original training), our evaluation process, and whole communication time. Their time costs are about 0.48 seconds, 1.25 seconds, and 10.57 seconds. Although our evaluation takes several times on time cost when compared with the original one, it only occupies a tiny part of the overall time (communication takes the most). Thus the evaluation process is far from being the bottleneck of the system. Besides, the evaluation process was only conducted in the several previous epochs in training, further releasing its computation burden.

VI. Related work

A. Federated Learning

Federated learning has been proposed to enable joint learning among distributed data owners without privacy leaking concerns [14]. Due to its great promise, substantial growth has occurred in this research field. For example, to address data non-IID issues, Zhao et al. [24] explain the impact with mathematical and try to release the problems by sanding a set of uniform distribution data among servers. Mehryar et al. [25] proposed an agnostic federated learning scheme to avoid the distribution bias. Recently, the DRL algorithm has been adapted to dynamically select a subset of training participants by Wang et al. [26], which can accelerate the model converge and alleviate the non-IID issue.

B. Graph Convolutional Networks

After been proposed, GCN shows its effectiveness in various fields like recommendation systems [17], temporal link prediction [27], and spam review detection [28]. A realistic situation is that the graph's size is usually too large to load in the memory. Thus various sampling strategies have been proposed to make the training process more efficient. GraphSAGE [29] mandates each node to sample a fixed number of neighbors in each layer, which may lead to data exploding when layers get deeper. To release the issue, VR-GCN [30] utilizes history neighbors' embedding to control variance and reduce the sampled neighbors to two nodes. Cluster-GCN [31] clusters the large graph into subgraphs and then completes convolution separately. FastGCN [32] fixes the number of nodes in each convolutional layer to avoid data exploding.

VII. Conclusion

This paper proposes a decentralized and secure federated graph learning system, named S-Glint, to jointly train a global GCN model among distributed graph data owners. We adopt homomorphic encryption (HE) based security protocol with pre-aggregation and batching strategies to preserve privacy efficiently. The traffic throttling and flow scheduling strategies are jointly optimized to alleviate the communication burden further. The former evaluates the embedding contribution of neighbors and selects partial of them while estimating others. The latter dynamic tuns the priority of flows according to the training stage and training completion time. We conducted multi-dimensional comparison experiments with various baselines, including a centralized solution and a decentralized solution. The experimental results have shown the superiority of S-Glint over the baselines.

References

[1] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[3] T. Liu, P. Li, and Y. Gu, "Glint: Decentralized federated graph learning with traffic throttling and flow scheduling," in *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*. IEEE, 2021, pp. 1–10.

[4] F. Chen, P. Li, T. Miyazaki, and C. Wu, "Fedgraph: Federated graph learning with intelligent sampling," *IEEE Transactions on Parallel Distributed Systems*, vol. 33, no. 08, pp. 1775–1786, aug 2022.

[5] H. H. Yang, Z. Liu, T. Q. Quek, and H. V. Poor, "Scheduling policies for federated learning in wireless networks," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 317–333, 2019.

[6] Y. Zhan, P. Li, and S. Guo, "Experience-driven computational resource allocation of federated learning by deep reinforcement learning," in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2020, pp. 234–243.

[7] Z. Li, H. Zhou, T. Zhou, H. Yu, Z. Xu, and G. Sun, "Esync: Accelerating intra-domain federated learning in heterogeneous data centers," *IEEE Transactions on Services Computing*, pp. 1–1, 2020.

[8] S.-J. Yang and Y.-R. Chen, "Design adaptive task allocation scheduler to improve mapreduce performance in heterogeneous clouds," *Journal of Network and Computer Applications*, vol. 57, pp. 61–70, 2015.

[9] J.-w. Lee, G. Jang, H. Jung, J.-G. Lee, and U. Lee, "Maximizing mapreduce job speed and reliability in the mobile cloud by optimizing task allocation," *Pervasive and Mobile Computing*, vol. 60, p. 101082, 2019.

[10] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pfabric: Minimal near-optimal datacenter transport," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 435–446, 2013.

[11] M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with varys," in *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014, pp. 443–454.

[12] H. Zhang, L. Chen, B. Yi, K. Chen, M. Chowdhury, and Y. Geng, "Coda: Toward automatically identifying and scheduling coflows in the dark," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 160–173.

[13] H. Tan, S. H.-C. Jiang, Y. Li, X.-Y. Li, C. Zhang, Z. Han, and F. C. M. Lau, "Joint online coflow routing and scheduling in data center networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 5, pp. 1771–1786, 2019.

[14] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the Artificial Intelligence and Statistics Conference (AISTATS)*, 2017, pp. 1273–1282.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 11,2022 at 08:19:27 UTC from IEEE Xplore. Restrictions apply.

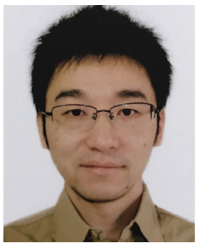
This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2022.3187149

[15] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–35, 2018.
[16] F. Chen, P. Li, D. Zeng, and S. Guo, "Edge-assisted short video sharing with guaranteed quality-of-experience," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2021.
[17] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 974–983.
[18] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning," in *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, 2020, pp. 493–506.
[19] T. Nishio, R. Shinkuma, and N. B. Mandayam, "Estimation of individual device contributions for incentivizing federated learning," *arXiv preprint arXiv:2009.09371*, 2020.
[20] M. Wang, L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, M. Li, J. Zhou, Q. Huang, C. Ma *et al.*, "Deep graph library: Towards efficient and scalable deep learning on graphs," *arXiv preprint arXiv:1909.01315*, 2019.
[21] https://snap.stanford.edu/data/p2p-Gnutella08.html.
[22] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," *arXiv preprint arXiv:1908.07782*, 2019.
[23] Z. Lin, C. Li, Y. Miao, Y. Liu, and Y. Xu, "Pagraph: Scaling gnn training on large graphs via computation-aware caching," in *Proceedings of the 11th ACM Symposium on Cloud Computing*, 2020, pp. 401–415.
[24] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.
[25] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," *arXiv preprint arXiv:1902.00146*, 2019.
[26] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1698–1707.
[27] K. Lei, M. Qin, B. Bai, G. Zhang, and M. Yang, "Gcn-gan: A non-linear temporal link prediction model for weighted dynamic networks," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 388–396.
[28] A. Li, Z. Qin, R. Liu, Y. Yang, and D. Li, "Spam review detection with graph convolutional networks," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 2703–2711.
[29] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in neural information processing systems*, 2017, pp. 1024–1034.
[30] J. Chen, J. Zhu, and L. Song, "Stochastic training of graph convolutional networks with variance reduction," *arXiv preprint arXiv:1710.10568*, 2017.
[31] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 257–266.
[32] J. Chen, T. Ma, and C. Xiao, "Fastgcn: fast learning with graph convolutional networks via importance sampling," *arXiv preprint arXiv:1801.10247*, 2018.

Tao Liu is a PhD student in the Graduate School of Computer Science and Engineering, The University of Aizu, Japan. His research interests mainly focus on cloud/edge computing, and distributed machine learning systems.

Peng Li received his BS degree from Huazhong University of Science and Technology, China, in 2007, the MS and PhD degrees from the University of Aizu, Japan, in 2009 and 2012, respectively. Dr. Li is currently an Associate Professor in the University of Aizu, Japan. His research interests mainly focus on cloud/edge computing, Internet-of-Things, machine learning systems, as well as related wired and wireless networking problems. Dr. Li has published over 100 technical papers on prestigious journals and conferences. He won the Young Author Award of IEEE Computer Society Japan Chapter in 2014. He won the Best Paper Award of IEEE TrustCom 2016. He supervised students to win the First Prize of IEEE ComSoc Student Competition in 2016. Dr. Li is the editor of IEICE Transactions on Communications, and IEEE Open Journal of the Computer Society. He is a senior member of IEEE.

Yu Gu received the B.E. degree from the Special Classes for the Gifted Young, University of Science and Technology of China, Hefei, China, in 2004, and the D.E. degree from the same university in 2010.In 2006, he was an Intern with Microsoft Research Asia, Beijing, China, for seven months. From 2007 to 2008, he was a Visiting Scholar with the University of Tsukuba, Tsukuba, Japan. From 2010 to 2012, he was a JSPS Research Fellow with the National Institute of Informatics, Tokyo, Japan. He is currently a Professor with the School of Computer and Information, Hefei University of Technology, Hefei, China. His current research interests include pervasive computing and affective computing. He was the recipient of the IEEE Scalcom2009 Excellent Paper Award and NLP-KE2017 Best Paper Award. He is a member of ACM and a senior member of IEEE.



Zhou Su received the Ph.D. degree from Waseda University, Tokyo, Japan, in 2003. His research interests include multimedia communication, wireless communication and network traffic. He is an Associate Editor for IET Communications, and an Associate Editor for the IEICE Transaction on Communications. He is the Chair of the Multimedia Services and Applications over Emerging Networks Interest Group (MENIG) of the IEEE Comsoc Society, the Multimedia Communications Technical Committee. He was the Co-Chair of several international conferences including IEEE VTC Spring 2016, IEEE CCNC2011, etc. He is a TPC Member of some flagship conferences including IEEE INFOCOM, IEEE ICC, IEEE Globecom, etc. He was the recipient of the best paper award of IEEE CyberScihech 2017, WiCon2016, and Funai Information Technology Award for Young Researchers.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 11,2022 at 08:19:27 UTC from IEEE Xplore. Restrictions apply.