2024 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA) | 979-8-3315-0971-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISPA63168.2024.00016

A Blockchain-Assisted Federated Learning Method for Recommendation Systems

Hang Zhang, Hao Tian, Cheng Chen, Chen Tian, and Wanchun Dou*
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
{hang.zhang, htian, cheng.chen}@smail.nju.edu.cn, {tianchen, douwc}@nju.edu.cn

***Abstract*— Due to the privacy advantages of federated learning (FL), federated recommendation systems (FedRSs) are gaining popularity for improving recommendation performance through training on local data. However, FedRSs frequently face the significant challenge of high communication costs between the server and clients. Most FedRSs utilize a client-server communication architecture, leading to heavy communication loads and single points of failure due to dependence on a central server. Clients may also encounter problems due to limited communication resources. In view of this challenge, in this paper, we propose a blockchain-assisted federated learning method at edge for communication-efficient recommendation systems, named BFedRec. Specifically, BFedRec reduces reliance on the central server by utilizing blockchain systems on edge servers to aggregate and distribute the recommendation model. To mitigate the high communication costs between clients and blockchain in each iteration, a communication-efficient training algorithm is used that trains the recommendation model directly on low-rank compressed parameters. Finally, we conduct extensive experiments on real-world datasets to verify the communication efficiency of BFedRec compared to existing methods. The experimental results show that BFedRec effectively improves communication efficiency without compromising recommendation performance.**

***Index Terms*—Federated Learning, Recommendation System, Communication Efficiency, Blockchain, Low-rank Training**

I. Introduction

With the rapid growth of e-commerce and digital services, digitization has become deeply ingrained in people's daily lives. Nowadays, individuals spend a significant portion of their time online, exploring a vast array of products, content, and services tailored to their specific interests. Recommendation systems play a crucial role for e-commerce [1] and digital service [2] providers by leveraging user data to offer personalized recommendations, thereby enhancing the overall user experience. These systems [3]–[5] analyze various types of data, such as browsing history, purchase records, and user interactions, to understand individual preferences and suggest relevant items or content.

While recommendation systems provide significant convenience and personalized user experiences, they also raise substantial privacy concerns. The collection and processing of user data, including sensitive information such as browsing history and purchase records, inherently pose risks of privacy breaches. Unauthorized access to this data can lead to severe security threats and potential misuse by malicious entities, putting user privacy. Moreover, there is a mounting apprehension that certain platforms might exploit user data for excessive and intrusive personalized recommendations, potentially violating users' privacy rights and diminish their trust in digital services.

Federated learning (FL) [6] has emerged as a revolutionary distributed machine learning framework that addresses privacy concerns associated with centralized data processing. In FL, deep learning models are collaboratively trained across multiple clients or devices, protecting data privacy by keeping user data on local devices instead of centralized servers. The development of federated recommendation systems (FedRSs) [7]–[10] is a direct extension of the principles of FL. In FedRSs, user data is securely stored and processed locally, with only model parameters uploaded for aggregation, ensuring data privacy and security.

However, conventional FedRSs assume ideal communication conditions between servers and clients, which is often not the reality. In practice, clients and servers commonly encounter communication bottlenecks like bandwidth limitations and high loads, significantly affecting system efficiency and performance.

- **High communication load on the server** is a common issue in FedRSs, especially when using large neural network models in modern recommendation systems. These models require frequent data exchanges for centralized aggregation, leading to servers handling multiple model updates and parameter synchronizations. This strains servers and increases network bandwidth requirements, exacerbating communication bottleneck issues. FL in a decentralized manner may be a feasible solution. Research on decentralized FedRSs is limited despite some progress in the field. Blockchain technology, with its security features like immutability, transparency, and traceability, offers a promising approach for decentralized aggregation and eliminating single points of failure in FedRSs. These features enhance system security and reliability, showcasing blockchain's potential in enhancing FedRSs.
- **Limited client-side communication resources** in FedRSs pose a significant challenge. Clients, often situated at the network edge with restricted bandwidth, face bottlenecks due to frequent data transmission and large model updates. Each server communication consumes

*Corresponding author.

2158-9208/24/$31.00 ©2024 IEEE
DOI 10.1109/ISPA63168.2024.00016

substantial network resources, causing delays and overall performance issues. Despite various technologies aimed at enhancing communication efficiency, the outcomes are suboptimal. Top-K compression minimizes the size of transmitted parameters by sparsification, yet clients incur extra expenses to organize and recognize updated data. Singular Value Decomposition (SVD) compression transmits low-rank model updates via singular value decomposition, but these updates may lose their low-rank status after server-side aggregation, ultimately failing to decrease downlink communication costs.

To address these issues, we propose BFedRec, a blockchain-assisted federated edge learning method with low-rank training for recommendation systems. Inspired by BEFL [11], we decrease dependence on central servers by implementing a blockchain network on edge nodes. By aggregating models through edge servers and propagating the global model via blockchain consensus mechanisms, decentralized aggregation and distribution of model have been achieved. To reduce communication costs between blockchain and clients during iterative training, we adopt a method that optimizes the model directly on low-rank parameters. It involves clients training the model on parameters with a low-rank structure and transmitting only the components of model updates between the client and the blockchain. Unlike SVD compression, it decreases communication load on both the uplink and downlink, while maintaining the performance and privacy of the recommendation system. The main contributions of this paper are as follows:

- We present a novel blockchain-based federated recommendation system for decentralized aggregation of recommendation models. It uses a blockchain network on edge nodes to replace the central server for global model aggregation and distribution.
- Inspired by SVD compression, we aim to optimize the model by focusing on low-rank structural parameters to reduce communication costs in both the uplink and downlink channels.

II. Preliminaries

In this section, we will introduce the preparation work and background of this paper. Additionally, we will also introduce knowledge of blockchain.

A. Federated Learning for Recommendation System

In a typical item-based FedRS [12], there are M users and N items. Each user u has a private interaction set $O_u = \{(i, r_{ui})\}$, where r_{ui} represents a rating or interaction value. Users aim to collaboratively construct a recommendation system while maintaining privacy. This situation is ideal for horizontal federated learning, where each user is an active participant.

The system aims to generate a list of the top K items that are both relevant to the user's preferences and have not been interacted with yet. This problem can be mathematically defined as finding a global model represented by parameters θ to minimize the global loss function $\mathcal{L}(\cdot)$.

$$\mathcal{L}(\theta) := \sum_{u=1}^{M} w_u \mathcal{L}_u(\theta) \tag{1}$$

Here, θ is the global parameter, w_u is the relative weight of user u, and $\mathcal{L}_u(\cdot)$ is the local loss function on user u's device. To achieve our objective, we utilize a federated matrix factorization (FedMF) [7] model as the core model in this paper, with the local loss function for user u defined as follows:

$$\begin{aligned}\mathcal{L}_u(p_u, Q) := &\sum_{(i,r_{ui})\in O_u} \ell\left(r_{ui}, (Q^T p_u)_i\right) \\ &+ \frac{\lambda}{2}\|p_u\|_2^2 + \frac{\lambda}{2}\|Q\|_2^2\end{aligned} \tag{2}$$

In the model, the local parameters of user u consist of the user embedding vector p_u and the item embedding matrix Q. The loss function $\ell(\cdot)$ quantifies the difference between predicted and actual values, while λ is the regularization parameter used to prevent overfitting.

Federated Averaging (FedAvg) [6] is one of the most popular algorithms in federated learning. It divides the training process into multiple rounds. At the beginning of each round, the server distributes the current item embedding matrix $Q^{(t)}$ to a group of users $\mathcal{S}(t)$, and then each user performs local SGD updates on their datasets. The local update rule for user u is as follows:

- user embedding vector p_u:

$$p_u^{(t+1)} = p_u^{(t)}(1 - \eta\lambda) + \eta {Q^{(t)}}^T (r - \hat{r}) \tag{3}$$

- item embedding matrix Q:

$$Q^{(t+1)} = Q^{(t)} - \eta(\lambda Q^{(t)} - (m * (r_u - \hat{r}_u)){p_u^{(t)}}^T) \tag{4}$$

Next, the user will send the local item embedding matrix update $\Delta_u^{(t)} = Q^{(t+1)} - Q^{(t)}$ to the server, and the server will perform aggregation steps to update the global model:

$$\theta^{(t+1)} = \theta^{(t)} + \frac{\sum_{u\in\mathcal{S}(t)} w_u \Delta_u^{(t)}}{\sum_{u\in\mathcal{S}(t)} w_u} \tag{5}$$

This process is repeated until the algorithm converges.

B. Blockchain

Blockchain is an innovative distributed ledger technology known for being tamper-proof, transparent, and decentralized, making it effective for addressing security issues in edge computing. By sharing a distributed ledger, participants can ensure the immutability, auditability, and verifiability of system transactions, enhancing security in traditional edge computing systems. Here are some key points about blockchain:

- **Nodes**: The blockchain network consists of multiple nodes, typically categorized into two types: user nodes, responsible for initiating transactions, and miner nodes, which maintain the blockchain database.

- **Transactions**: A transaction is a request to update the blockchain ledger by packaging data into a new block and appending it to the ledger.
- **IPFS (InterPlanetary File System)**: IPFS is a peer-to-peer distributed file system that provides permanent decentralized data storage. Data is stored in IPFS nodes and retrieved using a unique hash index derived from the data, which can be further secured with the tamper-resistant features of blockchain technology.
- **Consensus Mechanism**: Blockchain achieves consensus among decentralized entities through mechanisms such as Proof of Work and Proof of Stake, eliminating the need for centralized authority. These mechanisms ensure system security, transaction immutability, auditability, and verifiability.

III. A Blockchain-assisted Federated Learning Method for Recommendation Systems

In this section, we will introduce the proposed method by outlining its framework and explaining the specific tasks of different participants.

A. Overview

The framework of the BFedRec is illustrated in the Fig. 1. The BFedRec involves two main types of participants: user clients and a group of edge servers deploying blockchain, aiming to achieve communication-efficient federated recommendation, which are described as follows:

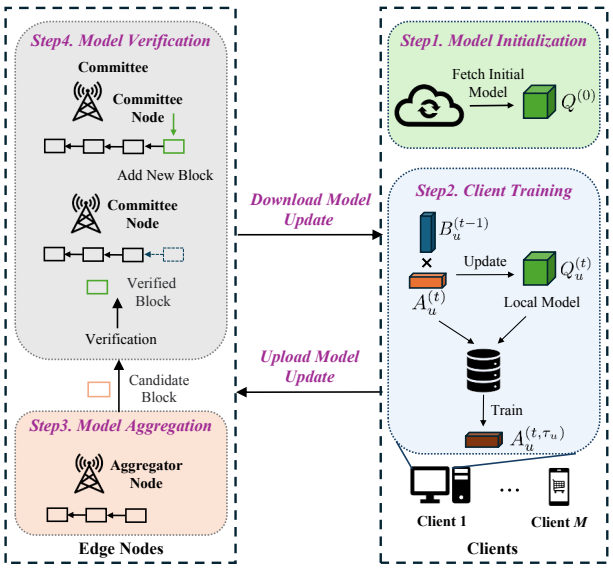

Fig. 1. The framework of BFedRec.

- **Clients**: Users interact with the blockchain through their clients, which collect local user data to train recommendation models. Before iterative training, the central server transfers the initial model to the blockchain network. Clients retrieve the initial model from the blockchain to initialize their local models. This will be detailed in the Sect. III-B. In a round of Fl, the client downloads the latest global model update from the blockchain, uses local private data for training, and uploads the trained update to the blockchain. In the training process, we uses low-rank structures to decompose the recommendation model update, reducing communication load. Further details will be discussed in Sect. III-C.
- **Edge Servers**: In order to address the high communication load issue caused by central server centralized aggregation of global models, we are considering using multiple edge servers to replace the central server, achieving decentralized model aggregation. A group of eligible edge servers serves as blockchain nodes, maintaining the blockchain system for recommendation model aggregation and verification. This will be detailed in the Sect. III-D.

B. Model Initialization

To address challenges like high communication load and single point of failure associated with the central server in FL, model aggregation is shifted to blockchain nodes (i.e., edge servers). Initially, the central server shares the IPFS address of the initial model and recommended task parameters (e.g., hyperparameters for FL training) to the blockchain network for federated training. After this phase, the central server is no longer needed for the FL process.

During iterative training rounds, clients request information from the nearest active blockchain node to get the IPFS address of the global model. Locally trained model updates are aggregated by decentralized edge nodes and the blockchain, distributing computational load and improving system robustness and security by removing the single point of failure of a central server.

By utilizing blockchain technology, the decentralized framework ensures the continuous and reliable nature of the FL process. Clients can easily access the latest global model and contribute their local updates, which are securely aggregated by the blockchain network. This distributed approach not only reduces communication bottlenecks but also utilizes the security and fault tolerance features of blockchain, offering a more scalable and resilient solution for FedRS.

C. Client Training

In FedRSs, communication costs of recommendation models are primarily influenced by updates to the item embedding matrix. To address the limited computational and communication resources for user clients, an efficient compression algorithm is essential to reduce data uploads and reduce communication costs.

As demonstrated in Equation 4, the update on the item embedding matrix consists of a rank-1 matrix and a regularization term. Inspired by [13], due to the small value of the regularization parameter λ, we restrict the updates of the

item embedding matrix to a low-rank structure to minimize communication costs. In particular, the local model update $\Delta_u^{(t)}$ is parameterized as:

$$\Delta_u^{(t)} = B_u^{(t)} A_u^{(t)} \tag{6}$$

where $B_u^{(t)} \in \mathbb{R}^{d \times r}$ and $A_u^{(t)} \in \mathbb{R}^{r \times N}$. Given that $r < N, d$, this approach reduces communication by a factor of $\frac{N \times d}{N \times r + r \times d}$.

To minimize downlink communication costs, only $A^{(t)}$ is transmitted between the client and the blockchain system. The blockchain aggregates components of model update to maintain low-rank downlink transmission. Each client's $B_u^{(t)}$ is updated before local training begins in each iteration.

Specifically, during the first round of training, clients obtain the initial global model $Q^{(0)}$ from the blockchain and use it to initialize their local models. Once clients receive the initial model, they can start their local training process. In each subsequent round of training, the server aggregates the local models $\{A_u^{(t-1,\tau_u)}\}$ uploaded by all clients to generate the updated global model $A^{(t)}$. This updated global model $A^{(t)}$ is then transmitted back to the clients.Upon receiving $A^{(t)}$, the clients use it to update their local models $Q_u^{(t)}$. Specifically, the clients merge their local models $Q_u^{(t)}$ with the global update $A^{(t)}$ to form a new $Q_u^{(t)}$. Next, they extract the updated $B_u^{(t)}$ from $Q_u^{(t)}$ and keep this $B_u^{(t)}$ fixed in the subsequent local training. At the beginning of each local training round, the clients initialize a new matrix $A_u^{(t,0)} = 0$. They then proceed with local training to optimize the local model parameters $\theta_u^{(t,0)} = \{A_u^{(t,0)}, p_u^{(t,0)}\}$. During local training, clients perform multiple updates based on their data and model parameters, gradually optimizing the local model. Once the training is complete, the clients upload the updated local model parameters $\{A_u^{(t,\tau_u)}\}$ to the server. After receiving the local model updates from all clients, the server aggregates them again to start a new round of global model updates. Through this iterative process, the system can continuously optimize both local models while maintaining efficient communication. Alg. 1 details the process of low-rank training.

Algorithm 1: Client Training

Input: Initial item embedding matrix $Q^{(0)}$ or latest model update $A^{(t)}$; current round t; rank r; learning rate of client η;
Output: The local model updates $\{A_1^{(t,\tau_u)}, \ldots, A_M^{(t,\tau_u)}\}$;

Sample a subset $\mathcal{S}^{(t)}$ of clients;
for ***each*** *client* $u \in \mathcal{S}^{(t)}$ ***in parallel*** **do**
  if $t = 0$ **then**
    Download $Q^{(0)}$;
    Initialize $Q_u^{(t)} = Q^{(0)}$;
  else
    Download $A^{(t)}$;
    Merge $Q_u^{(t)} = Q_u^{(t-1)} + B_u^{(t-1)} A^{(t)}$;
  Update $B_u^{(t)}$ from $Q_u^{(t)}$;
  Initialize $Q_u^{(t,0)} = Q_u^{(t)}$ and $A_u^{(t,0)} = 0$;
  Set trainable parameters $\theta_u^{(t,0)} = \{A_u^{(t,0)}, p_u^{(t,0)}\}$;
  for $k = 0, \ldots, \tau_u - 1$ **do**
    Perform local update $\theta_u^{(t,k+1)} = \mathrm{GSD}\left(\theta_u^{(t,k)}, \nabla \mathcal{L}_u\left(\theta_u^{(t,k)}\right), \eta\right)$;
  $p_u^{(t+1)} = p_u^{(t,\tau_u)}$;
  Upload $A_u^{(t,\tau_u)}$ to the nearest blockchain node;

D. Model Aggregation and Verification

We deploy a blockchain network through a set of edge servers. Each blockchain node is assigned a Verifiable Random Functions (VRF) key pair for committee composition, a signature key pair, and default stake. The blockchain comprises blocks linked by hash pointers, with each block header containing an index, timestamp, and the hash value of the previous block. The block body of a candidate block includes multiple transactions representing local model updates and the IPFS address of the global model. Once a candidate block is confirmed, local model updates are removed, and the IPFS address of the global model published by the committee is recorded in the block body.

During the collection phase of model updates, each blockchain node receives model update transactions from clients. Nodes verify signatures and updates before propagating the transactions to the blockchain network and adding them to the blockchain. The latest block then includes multiple model updates and client signatures. Once a blockchain node collects enough model update transactions, it competes to calculate a new global model update and records its IPFS address in a block to form a candidate block. This block is then sent to a committee of multiple nodes for model verification.

During model verification phase, committee nodes verify the correctness of the global model update aggregated from aggregator node through voting. If a candidate block receives over two-thirds of the approval votes from the committee, it becomes a validated block and is shared with the entire network. This validated block replaces the original local model updates and includes the committee nodes' signatures. Upon receiving the validated block, other blockchain nodes will verify the committee nodes' signatures and add it to their local chain. Other candidate blocks in this round will be invalidated and cleared. If sufficient consensus is not obtained within the specified time, the committee will discard the candidate block and wait for the next one. If no candidate block is confirmed in multiple rounds of voting, a new committee will be established. The process of blockchain aggregation and verification is detailed in Alg. 2

By using VRF, our method selects committee members in a private and non-interactive way. We introduce a novel sortition algorithm based on VRF to choose a random subset of nodes

Algorithm 2: Model Aggregation and Verification

Input: The received model updates $\{A_1^{(t,\tau_u)}, \ldots, A_M^{(t,\tau_u)}\}$; the maximum vote round $MaxStep$, maximum voting time $MaxVoteDuration$;

Output: The verified block $vBlock$;

```
for each blockchain node do
    Collect received model updates;
    Validate updates and add to the candidate block cBlock;
    if updates are sufficient then
        Aggregate updates A^(t+1) = Σ_{u∈S^(t)} (N_u/N) A_u^(t,τ_u);
        Record A^(t+1) to cBlock and send it to committee;
while True do
    Initialize the current voting round step = 1;
    while step < MaxStep do
        if voting time < MaxVoteDuration then
            Committee nodes vote for cBlock;
            if agreements more than 2/3 then
                The cBlock is verified successfully, and form vBlock;
            else
                Verification of the cBlock failed;
                step++ and wait for a new candidate block;
    Committee reconstruction;
```

for the committee. Each node independently runs the sortition algorithm with a public seed (i.e. the latest blockchain block) to confirm their selection as a candidate committee member. The sortition algorithm selects nodes based on their equity proportion. If a unique random hash generated from a key and input seed meets the criteria, the node becomes a candidate committee member and shares qualifying information with the network (including hash and proof). The committee formation phase concludes when the predetermined committee size is reached, with initial members becoming authoritative. Any node can access committee information from the network and verify committee members' identities using the public key of VRF, the latest blockchain block, and their equity holdings.

E. Summary of BFedRec

As shown in Fig. 1, we use blockchain to share the workload of the server and enhance communication efficiency of the system. The summarized workflow of the BFedRec includes the following steps, as illustrated in Alg. 3.

1) The central server first publishes FL tasks and the initial recommendation model stored in IPFS to the blockchain.

Algorithm 3: BFedRec

```
Server publishes an initial model to the blockchain;
User retrieves IPFS address of global model from the nearest blockchain node and downloads the model from IPFS;
for t = 0, 1, 2, ..., T do
    User download the last global model update from the blockchain;
    User updates model locally using private data by Alg. 1 and sends the update to the nearest blockchain node;
    Blockchain aggregates and verifies updates to form a new verified block by Alg. 2;
    Nodes add the verified block to their local chain.
```

2) The user retrieves the IPFS address of the initial model by querying the nearest active blockchain node and downloads the model using this address.
3) Users use their private data for local model training. A low-rank structure is adopted to decompose the item embedding matrix for model compression.
4) After local training, the user sends the compressed model update and signature to the nearest blockchain node as a transaction.
5) Upon receiving the model update from the user, the blockchain node verifies the signature and the update. After verified, the node shares the transaction across the blockchain network. Once a sufficient number of model update transactions are gathered, nodes will compete to calculate a new global model update to generate a candidate block, which are subsequently transmitted to the committee. The committee validates the global model through voting and propagates the verified block to the blockchain network.
6) All blockchain nodes add the verified block to their local chain. The current training round is finished, and it will proceed to the next round until the model converges.

IV. Experiments

In this section, we conduct experiments to evaluate the performance of our proposed method. Our experiments seek to answer the following research questions:

- **RQ1:** How does our method compare to the benchmark in terms of recommendation performance?
- **RQ2:** How is our method performing in terms of improving communication efficiency?
- **RQ3:** How does our method demonstrate Byzantine fault tolerance against malicious blockchain nodes?

A. Experimental Setup

1) Datasets: We evaluated the our on three widely used real-world datasets: MovieLens-1M [14], Pinterest [12], and LastFM-2K [15]. Table I summarizes the characteristics of

these datasets. We retain users with at least 20 interactions and convert numerical ratings into implicit feedback.

TABLE I
STATICS OF THE DATASETS.

Statics / Dataset	Users	Items	Interactions	Sparsity
ML-1M	6,040	3,706	1,000,209	95.53%
Pinterest	55,187	9,916	1,500,809	99.73%
LFM-2K	1,600	12,454	185,650	99.07%

2) Baselines: We will experimentally compare our method with federated recommendation methods and compression competitors:

- **Federated Recommendation methods:**
  - **FedMF** [7]: is one of the earliest implicit feedback collaborative filtering methods based on the federated learning paradigm.
  - **FedRec** [8]: a method that uploads the set of items that users have interacted with and a random sample of other items to achieve privacy preservation during rating prediction tasks.
- **Compression competitors:**
  - **Sparsification**: It refers to setting some parameters in the model so that the model updates become sparse representing the updates as sparse matrices. Here, we use a random selection of K variables for sparse representation.
  - **Top-K Compression**: It is based on sparsification where only the most important K values are transmitted instead of updates to reduce the transmission size.
  - **SVD Compression**: It returns low-rank structured compressed updates based on singular value decomposition.

3) Evaluation Metrics: To evaluate the performance of our method in predicting interactions, we choose Hit Rate (HR) [16] and Normalized Discounted Cumulative Gain (NDCG) [17] as evaluation metrics. HR indicates whether the test items appear in the recommended list, while NDCG assigns higher weights to items ranked higher, thereby measuring the position of the hit items. In this paper, we set the length L of recommendation list to 10. We calculate these evaluation metrics for each client and report the average scores to assess the recommendation performance comprehensively.

4) Implementation Details: For the base model, we employ the matrix factorization algorithm. In the experiment, the embedding dimensions for users and items in the MovieLens-1M dataset are set to 64, while for the Pinterest dataset, the embedding dimensions are set to 16. We use a simple SGD optimizer for local training on edge devices. In each round, we randomly sample M clients without replacement within and across rounds. E rounds of training are conducted on the dataset on each client.

B. Performance Analysis (RQ1)

In performance analysis, we mainly focus on the recommended performance in communication-constrained environments. We compare the recommended performance of the system proposed in this paper with FedMF and FedRec under the same communication budget (i.e. embedding matrices of the same size). On the ML-1M dataset, we set the dimension of transmitted item embeddings to 4, while the embedding size of BFedRec is fixed at 64. Similarly, for the Pinterest dataset, the item embedding size of FedMF is 1, while BFedRec has an embedding size of 16. Our setup ensures that the transmission sizes of the two methods are roughly equal in each dataset.

In Table II, we show the HR and NDCG metrics for different methods. With the same communication budget, BFedRec outperforms FedMF and FedRec on both datasets. On the Pinterest dataset, even with an update size of only 1/16 of the maximum model, BFedRec still achieves significant performance improvement (0.60 HR and 0.33 NDCG) compared to FedMF (0.41 HR and 0.22 NDCG) and FedRec (0.43 HR and 0.25 NDCG). In contrast, the accuracy of the FedMF model with the corresponding embedding size is much lower.

TABLE II
RECOMMENDATION PERFORMANCE.

Dataset	Metric	Centralized		Federated		
		BPR	GMF	FedMF	FedRec	BFedRec
ML-1M	HR	86%	61%	41%	43%	**60%**
	NDCG	29%	34%	22%	25%	**33%**
Pinterest	HR	89%	83%	23%	25%	**78%**
	NDCG	52%	49%	12%	15%	**45%**
LFM-2K	HR	90%	82%	68%	72%	**83%**
	NDCG	47%	65%	49%	50%	**53%**

In the MovieLens-1M and LastFM-2K datasets, we also observed similar patterns, where BFedRec consistently exhibited higher recommendation performance among similar methods. Experimental results indicate that compared to fully trained methods FedMF and FedRec, BFedRec can achieve competitive performance while significantly reducing communication costs.

Overall, the BFedRec method demonstrates higher recommendation performance in all test datasets, especially in communication-constrained environments, making it an efficient and competitive recommendation system solution.

C. Communication Efficiency Analysis (RQ2)

The Table III and Fig. 2 compare the communication time (measured in minutes) of four compression methods, i.e., Sparsification, Top-K compression, SVD compression, and BFedRec with different embedding sizes (1, 2, 4, 8, 16).

The communication time of the Sparsification method, which involves randomly selecting K variables for sparse representation, increases from 1.56 minutes to 31.26 minutes as the embedded size for transmission increases from 1 to 16. This indicates that the communication time for Sparsification significantly increases with the increase in embedded size.

TABLE III
COMPARISON OF COMPRESSION METHODS.

Method \ Size	1	2	4	8	16
Sparsification	1.56	3.12	5.98	16.72	31.26
Top-K Compression	2.51	5.03	10.05	20.11	40.21
SVD Compression	1.72	3.45	7.34	17.23	32.12
BFedRec	**1.26**	**2.51**	**5.03**	**10.05**	**20.11**

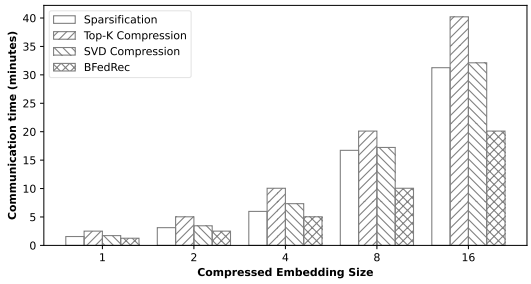

Fig. 2. Comparison of Compression Methods.

Similarly, the communication time of the Top-K compression method increases from 2.51 minutes to 40.21 minutes as the embedded size for transmission increases from 1 to 16. This demonstrates that the communication time of the Top-K method also significantly increases with the increase in embedded size. The communication time of the SVD compression method increases from 1.72 minutes to 32.12 minutes for the same embedded sizes for transmission. Compared to Top-K, the communication time of the SVD method is slightly lower for smaller embedded sizes, but becomes closer to Top-K for larger embedded sizes. The BFedRec method shows the most stable communication time across all embedded sizes, increasing from 1.26 minutes to 20.11 minutes. Although its communication time is slightly lower than Top-K and SVD for larger embedded sizes, it exhibits higher performance for smaller embedded sizes, indicating that BFedRec has an advantage in communication-constrained environments.

Overall, the BFedRec method performs better in communication time for smaller embedded sizes compared to the other three methods. As the embedded size increases, its communication time increases more steadily. This indicates that BFedRec can maintain high efficiency under different communication budgets, making it an efficient and stable compression method.

D. Security analysis (RQ3)

The security of BFedRec relies on a committee-based consensus protocol. However, fully validating its security through testing all potential attack strategies is impractical. Malicious blockchain nodes disrupt the system by influencing committee formation, candidate block generation, and voting processes. They cast dissenting votes for correct candidate blocks and approve votes for incorrect ones. Moreover, during candidate block generation, malicious nodes conduct Gaussian attacks by setting random values as global model parameters instead of adhering to the secure aggregation protocol.

In the experiment, one-third of the blockchain nodes (the maximum Byzantine fault tolerance in distributed systems) were considered malicious and colluded to mislead the learning process. The results in Fig. 3 show that the performance of BFedRec was not affected too much by these Byzantine nodes, demonstrating its robustness against this attack scenario.

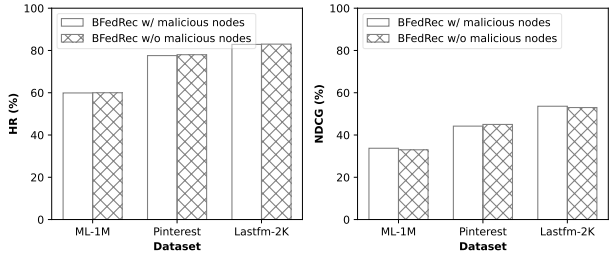

Fig. 3. Performance with vs. without Malicious Nodes.

V. RELATED WORK

A. Federated Recommendation Systems

FL has been widely applied to address privacy issues in recommendation systems. FCF [7] and FedRec [8] are examples of FL-based collaborative filtering approaches used to enhance privacy in rating prediction tasks. In addition, some researchers have proposed personalized federated recommendation frameworks. MetaMF [9] uses meta-learning to train personalized models for each client for more accurate recommendations. HPFL [10] introduces a hierarchical personalized federated learning framework to improve recommendation performance by adapting to data heterogeneity. In recent years, federated recommendation systems have advanced significantly in various fields. Researchers have proposed innovative approaches to improve accuracy, user experience, and security. Ye et al. [18] and He et al. [19] introduced clustering-based federated learning frameworks. Zhang et al. [20] addressed the cold start problem by separating item attributes and user interaction data. Nguyen et al. [13] optimized communication costs and security through low-rank training. Qu et al. [21] enhanced security and user control with personalized privacy protection. Yan et al. [22] developed a privacy-protected recommendation system using a federated heterogeneous graph neural network.

B. Blockchain-based Federated Learning

FL is vulnerable to single points of failure and requires a more distributed and secure approach to enable collaboration among participants. To address these issues, researchers have suggested integrating blockchain with FL. Nguyen et al. [23] proposed FedChain, a secure framework based on Proof of Stake (PoS) for federated blockchain systems, enhancing security and performance of cross-chain asset transfers. Wu et al.

[24] proposed a blockchain-based clustering federated learning system using clustering and distillation techniques for peer-to-peer knowledge transfer support. Jin et al. [11] presented the BEFL system, addressing issues like malicious clients and communication costs by utilizing lightweight blockchain technology. Meanwhile, various applications of blockchain-based FL exist. Guo et al. [25] achieved secure traffic prediction, improved data privacy, and model accuracy through B2SFL, a dual-layer blockchain architecture. Aloqaily et al. [26] proposed combining digital twins and blockchain-assisted federated learning to enhance Industry 4.0, improving the intelligence and security of manufacturing processes. Ayepah-Mensah et al. [27] introduced a blockchain-based federated learning resource allocation and trading system, enhancing the efficiency and privacy of 5G network slicing.

VI. Conclusion

In this paper, we present a blockchain-assisted federated edge learning method for recommendation systems (BFedRec) that utilizes blockchain technology on edge nodes to decentralize aggregation and distribution of recommendation models. A low-rank training method is employed to reduce communication costs between the blockchain and clients. Experimental results demonstrate enhanced communication efficiency without compromising recommendation performance.

Acknowledgment

This research is supported part by the National Key Research and Development Program of China (2022YFB2702803), National Natural Science Foundation of China (92267104).

References

[1] Y. Gong, Z. Jiang, Y. Feng, B. Hu, K. Zhao, Q. Liu, and W. Ou, "Edgerec: recommender system on edge in mobile taobao," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2477–2484.

[2] S. Wang, S. Guo, L. Wang, T. Liu, and H. Xu, "Hdnr: A hyperbolic-based debiased approach for personalized news recommendation," in *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023, pp. 259–268.

[3] Y. Hu, X. Xu, L. Duan, M. Bilal, Q. Wang, and W. Dou, "End-edge collaborative inference of convolutional fuzzy neural networks for big data-driven internet of things," *IEEE Transactions on Fuzzy Systems*, 2024.

[4] X. Xu, F. Wu, M. Bilal, X. Xia, W. Dou, L. Yao, and W. Zhong, "Xrl-shap-cache: an explainable reinforcement learning approach for intelligent edge service caching in content delivery networks," *Science China Information Sciences*, vol. 67, no. 7, p. 170303, 2024.

[5] X. Xu, H. Dong, L. Qi, X. Zhang, H. Xiang, X. Xia, Y. Xu, and W. Dou, "Cmclrec: cross-modal contrastive learning for user cold-start sequential recommendation," in *Proceeding of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2024.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[7] M. Ammad-Ud-Din, E. Ivannikova, S. A. Khan, W. Oyomno, Q. Fu, K. E. Tan, and A. Flanagan, "Federated collaborative filtering for privacy-preserving personalized recommendation system," *arXiv preprint arXiv:1901.09888*, 2019.

[8] G. Lin, F. Liang, W. Pan, and Z. Ming, "Fedrec: Federated recommendation with explicit feedback," *IEEE Intelligent Systems*, vol. 36, no. 5, pp. 21–30, 2020.

[9] Y. Lin, P. Ren, Z. Chen, Z. Ren, D. Yu, J. Ma, M. d. Rijke, and X. Cheng, "Meta matrix factorization for federated rating predictions," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 981–990.

[10] J. Wu, Q. Liu, Z. Huang, Y. Ning, H. Wang, E. Chen, J. Yi, and B. Zhou, "Hierarchical personalized federated learning for user modeling," in *Proceedings of the Web Conference 2021*, 2021, pp. 957–968.

[11] R. Jin, J. Hu, G. Min, and J. Mills, "Lightweight blockchain-empowered secure and efficient federated edge learning," *IEEE Transactions on Computers*, vol. 72, no. 11, pp. 3314–3325, 2023.

[12] D. Chai, L. Wang, K. Chen, and Q. Yang, "Secure federated matrix factorization," *IEEE Intelligent Systems*, vol. 36, no. 5, pp. 11–20, 2020.

[13] N.-H. Nguyen, T.-A. Nguyen, T. Nguyen, V. T. Hoang, D. D. Le, and K.-S. Wong, "Towards efficient communication and secure federated recommendation system via low-rank training," in *Proceedings of the ACM on Web Conference 2024*, 2024, pp. 3940–3951.

[14] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[15] I. Cantador, P. Brusilovsky, and T. Kuflik, "Second workshop on information heterogeneity and fusion in recommender systems (hetrec2011)," in *Proceedings of the fifth ACM conference on Recommender systems*, 2011, pp. 387–388.

[16] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 143–177, 2004.

[17] X. He, T. Chen, M.-Y. Kan, and X. Chen, "Trirank: Review-aware explainable recommendation by modeling aspects," in *Proceedings of the 24th ACM international on conference on information and knowledge management*, 2015, pp. 1661–1670.

[18] Z. Ye, X. Zhang, X. Chen, H. Xiong, and D. Yu, "Adaptive clustering based personalized federated learning framework for next poi recommendation with location noise," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 5, pp. 1843–1856, 2024.

[19] X. He, S. Liu, J. Keung, and J. He, "Co-clustering for federated recommender system," in *Proceedings of the ACM on Web Conference 2024*, 2024, pp. 3821–3832.

[20] C. Zhang, G. Long, T. Zhou, Z. Zhang, P. Yan, and B. Yang, "When federated recommendation meets cold-start problem: Separating item attributes and user interactions," in *Proceedings of the ACM on Web Conference 2024*, 2024, pp. 3632–3642.

[21] L. Qu, W. Yuan, R. Zheng, L. Cui, Y. Shi, and H. Yin, "Towards personalized privacy: User-governed data contribution for federated recommendation," *arXiv preprint arXiv:2401.17630*, 2024.

[22] B. Yan, Y. Cao, H. Wang, W. Yang, J. Du, and C. Shi, "Federated heterogeneous graph neural network for privacy-preserving recommendation," *arXiv preprint arXiv:2310.11730*, 2023.

[23] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, Y. Xiao, H.-A. Pham, E. Dutkiewicz, and N. H. Tuong, "Fedchain: Secure proof-of-stake-based framework for federated-blockchain systems," *IEEE Transactions on Services Computing*, vol. 16, no. 4, pp. 2642–2656, 2023.

[24] H. Wu, X. Zhu, and W. Hu, "A blockchain system for clustered federated learning with peer-to-peer knowledge transfer," *Proceedings of the VLDB Endowment*, vol. 17, no. 5, pp. 966–979, 2024.

[25] H. Guo, C. Meese, W. Li, C.-C. Shen, and M. Nejad, "B 2 sfl: A bi-level blockchained architecture for secure federated learning-based traffic prediction," *IEEE Transactions on Services Computing*, 2023.

[26] M. Aloqaily, I. Al Ridhawi, and S. Kanhere, "Reinforcing industry 4.0 with digital twins and blockchain-assisted federated learning," *IEEE Journal on Selected Areas in Communications*, 2023.

[27] D. Ayepah-Mensah, G. Sun, G. O. Boateng, S. Anokye, and G. Liu, "Blockchain-enabled federated learning-based resource allocation and trading for network slicing in 5g," *IEEE/ACM Transactions on Networking*, 2023.