2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

Cache Optimization in Consortium Blockchain System Based on GCN and XGBoost

Ao Xiong[1], Wenchuan Ma[1(✉)], Yan Zhang[2], Zhe Du[3], Xuwen Liu[4], He Huang[5]

[1] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2] State Grid Beijing Electric Power Company, Beijing 100051, China
[3] State Grid Digital Technology Holding Co., Ltd., Beijing 100032, China
[4] State Grid Jiangsu Electric Power Co., Ltd., Nanjing 210000, China
[5] State Grid Jiangsu Electric Power Co., Ltd., Changzhou 213003, China
mawenchuan@bupt.edu.cn

Abstract. With the rapid development of the global carbon market, the application of consortium blockchain technology in carbon trading and carbon neutrality management has become increasingly widespread, ensuring the security and transparency of transaction data. However, as the transaction scale continues to expand, the on-chain storage capacity and query efficiency of blockchain systems have gradually become key factors limiting system performance. Although traditional off-chain storage solutions have alleviated the pressure on on-chain storage to some extent, high-concurrency access scenarios still face the challenge of high query latency. While in-memory caching methods can improve query speed, they are often limited by memory capacity, making it difficult to meet the query response time requirements when dealing with large amounts of data. To address these issues, this paper proposes a cache optimization strategy based on transaction access prediction, combining Graph Convolutional Networks (GCN) and Extreme Gradient Boosting (XGBoost). The method first constructs a relationship graph of transaction data using GCN and extracts the structural features of transaction nodes. It then combines the XGBoost model to predict access frequency and dynamically adjusts the cache replacement strategy. Experimental results show that, compared to traditional algorithms, the proposed method substantially increases cache hit rates and reduces query latency under high-concurrency conditions. This study provides an intelligent optimization solution for cache management in blockchain trading systems, which is of great significance for improving the operational efficiency of the carbon trading market.

Keywords: Blockchain, Cache Optimization, Graph Convolutional Networks.

1 Introduction

As global carbon emission targets continue to progress, carbon trading and carbon neutrality management have gained increasing attention worldwide. Blockchain technology, with its characteristics of decentralization, immutability, and transparency, has become an important tool to ensure the trustworthy flow of data across various stages of

the carbon market. A consortium blockchain is a blockchain network maintained by multiple trusted entities, featuring permission control and data privacy protection, suitable for scenarios requiring collaboration and trust, and commonly used by enterprises or governments. Consortium blockchain can effectively record every transaction in carbon emission trading, ensuring the authenticity and transparency of carbon credit transactions while improving the efficiency and trust of carbon market trading. By utilizing consortium blockchain technology, each carbon trading transaction can be recorded in a distributed ledger in real-time and in an open manner, effectively addressing data tampering and trust issues that may arise during the trading process.

However, with the continued expansion of the carbon market, especially with the large-scale on-chain storage of carbon measurement data, the storage capacity and access efficiency of blockchain systems have become one of the bottlenecks limiting its application in carbon trading and carbon neutrality management. Traditional blockchain systems face the problem of limited storage space [1], and the on-chain storage capacity cannot support the large-scale uploading of carbon data, leading to access delays and low query efficiency, which in turn affects the overall operational efficiency of the carbon market. These storage and performance issues pose a significant challenge to the deep integration of blockchain technology with carbon trading and carbon neutrality management.

To alleviate the pressure on on-chain storage and improve performance, many researchers[1, 2, 3, 4] have proposed solutions to transfer large amounts of data to off-chain storage. Off-chain storage uses traditional database systems to store blockchain transaction data. This method reduces the storage burden on the blockchain and improves data processing efficiency. Especially in blockchain networks with limited storage capacity, off-chain storage provides greater flexibility and scalability. However, off-chain storage systems still face performance bottlenecks, such as high query latency, in high-concurrency access scenarios, particularly when dealing with large-scale transaction requests and uneven data access frequencies. Traditional databases often fail to meet the real-time and efficiency requirements.

Table 1. Comparison of Query Concurrency Performance

Name	Query Concurrency (Requests/sec)	Model
Hyperledger Fabric	100 - 1,000	Peer-to-peer, Consensus-based
MySQL	1,000 - 10,000	Multi-threaded, SQL-based
Redis	10,000-1,00,000	Single-threaded, Event-driven

As shown in Table 1, traditional consortium blockchain Hyperledger Fabric and traditional database MySQL exhibit low efficiency when handling high-concurrency queries [5]. Therefore, this paper proposes the introduction of a caching layer placed between the database and the user. Since the cache is memory-based, it can put significant pressure on memory when dealing with large amounts of transactional data. Existing

cache replacement strategies, such as Least Recently Used (LRU) and Least Frequently Used (LFU), although improving cache performance to some extent, still exhibit limitations in complex data interactions and dynamic access patterns. LRU relies on recent access records, while LFU replaces based on access frequency, but neither can effectively handle the dynamic changes in transaction data and diverse access patterns. Therefore, predicting the access frequency of transaction data and dynamically adjusting the cache strategy based on access frequency has become a key issue in optimizing blockchain transaction systems.

To address this issue, this paper proposes an efficient cache optimization technique based on the combination of Graph Convolutional Networks (GCN) and Extreme Gradient Boosting (XGBoost). By constructing a relationship graph model of transaction data, GCN captures the complex correlations between transaction data, enabling accurate prediction of access frequency. The GCN model aggregates feature information from neighboring nodes to discover potential associations and patterns, improving the accuracy of future access frequency predictions. Combined with the XGBoost model, the prediction accuracy is further enhanced, dynamically adjusting the cache replacement strategy. This prediction result not only facilitates intelligent cache replacement but also effectively improves cache hit rates and reduces query latency. Compared to traditional cache replacement strategies, the GCN and XGBoost-based method can effectively enhance system responsiveness in high-concurrency scenarios and avoid performance bottlenecks caused by uneven access frequencies.

2 Related Works

In recent years, blockchain storage scalability and architecture optimization have become hot research topics. Liu et al. [1] proposed dynamic sharding technology, which first achieved off-chain storage scalability, laying the foundation for distributed load balancing. Qian et al. [2] developed middleware for database tamper detection, enhancing off-chain data integrity through blockchain notarization. Many researchers improved the data consistency verification mechanism by integrating the IPFS protocol, significantly enhancing the reliability of distributed storage [3, 4, 6]. Maheshwarkar et al. [7] strengthened storage disaster recovery capabilities through a distributed redundancy architecture. Yang et al. [8] developed the ChainMaker Storage System (CMSS), which innovatively integrates block storage workflows with a meta-file system. Liu et al. [9] further designed an auxiliary storage solution using cache prefetching technology to optimize blockchain query efficiency.

In the field of index optimization and query enhancement, key technologies have made diverse breakthroughs. Zhou et al. [10] developed the MSTDB technology, which pioneered the Merkle Semantic Dictionary Tree, achieving secure and efficient linkage between on-chain and off-chain data. Rosoon et al. [11] proposed a dynamic trusted verification mechanism that achieves a dynamic balance between query speed and data trustworthiness in hybrid architectures. Pandita et al. [12] combined PCA and Isolation Forest algorithms to establish a real-time anomaly detection framework for blockchain databases. Wang et al. [13] designed a cooperative storage scheme based on a timeline

adaptive reading mechanism, optimizing the access performance of lightweight blockchains through redundant caching strategies. Cai et al. [14] proposed a dual-chain architecture with data type separation strategies, providing theoretical support for dynamic resource scheduling. Li et al. [15] developed the EtherQL abstraction query layer, which first supports advanced query functions for blockchain systems. El-Hindi et al. [16] validated a smart contract-driven sharding strategy, revealing the deep collaborative technological path between blockchains and distributed databases. Yu et al. [17] developed the FabricSQL system, which innovates relational query paradigms through encryption verification mechanisms, while Miyachi et al. [18] constructed a privacy-enhanced modular off-chain storage framework, promoting the application of blockchain in healthcare and other fields.

However, these methods still face some usability issues when handling high-concurrency access, making the introduction of off-chain caching necessary to improve blockchain's off-chain performance. Tu et al. [19] proposed an optimization scheme based on Redis caching and B+ tree indexing to address the retrieval efficiency bottleneck in consortium blockchains with multiple transaction modes. However, since Redis is an in-memory caching middleware, it may cause excessive memory pressure when dealing with large-scale transaction data, making it necessary to research an intelligent cache optimization algorithm in the blockchain domain.

In other fields, machine learning-based cache optimization methods have gradually gained attention. For example, Narayanan et al. [20] used LSTM encoder-decoder models to predict content popularity, improving traditional cache strategy performance. Shi et al. [21] designed a low-overhead Glider replacement strategy by lightweighting the LSTM model through interpretability analysis. Hou et al. [22] applied graph neural networks (GNN) to cache performance optimization in Named Data Networking (NDN), demonstrating its potential in modeling complex data correlations. Compared to the general GNN, Graph convolutional networks (GCN) [23], as a key branch of GNN, explicitly defines neighborhood information propagation mechanisms through graph convolutional layers, thereby making it more suitable for structured feature extraction. It has achieved significant results in fields like social network analysis and recommendation systems.

3 Proposed Method

3.1 System Architecture

This system addresses the high-concurrency transaction query challenge in consortium blockchain environments by constructing an efficient storage architecture comprising a blockchain layer, an off-chain database layer, and an off-chain cache layer, and by introducing a predictive mechanism that integrates GCN with XGBoost to optimize cache management strategies—thereby improving data access efficiency and reducing query latency.

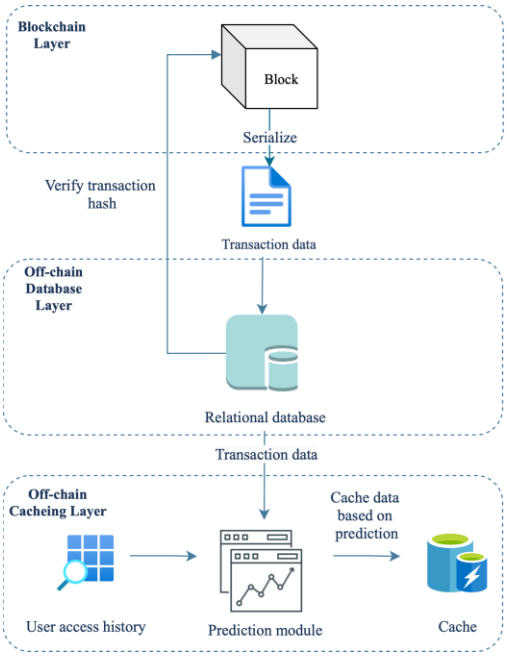

Fig. 1. System Architecture Diagram

Fig. 1 illustrates the three-tier storage and caching architecture. The blockchain layer serializes and packages each transaction validated by the consensus mechanism, storing only the transaction hash and essential metadata on-chain to guarantee data integrity and traceability via smart contracts. Beneath this, the off-chain database layer employs high-performance relational databases or distributed storage systems to store complete transaction details (including sender, receiver, amount, timestamp, etc.), supporting large-scale data queries and access but potentially suffering performance bottlenecks under surges in transaction volume, frequent access, and complex queries. To mitigate these bottlenecks, the off-chain cache layer is introduced above the database layer: it first analyzes users' transaction access histories and constructs a transaction relationship graph, then uses GCN to extract node embeddings that capture inter-transaction relationships and access patterns. These embeddings are concatenated with the original transaction features and fed into an XGBoost model to predict future access frequencies. Based on these predictions, the cache content is dynamically adjusted—high-frequency transactions are preferentially retained, while low-frequency transactions are evicted when cache space is constrained—thus effectively relieving database pressure and significantly reducing query latency.

This architecture maintains the security and immutability of transaction data while, through the efficient coordination of the blockchain, off-chain database, and off-chain cache layers, markedly enhancing query efficiency for high-frequency transactions. In scenarios involving large-scale transaction volumes and high-concurrency requests, it

effectively mitigates database query pressure, shortens access latency, and optimizes the user experience in complex business environments such as carbon trading.

3.2 Characteristics of Consortium Blockchain Transaction Data

In the consortium blockchain environment, transaction data exhibits characteristics that differ from those of public blockchains, which directly impact storage architecture, query methods, and cache management strategies. Permissioned blockchains are maintained by multiple trusted entities, such as enterprises, institutions, or government departments, and their transaction data access patterns and storage requirements are more regular compared to public blockchains. Additionally, there are higher demands for privacy protection and performance optimization. The access control of transaction data in permissioned blockchains ensures that data access patterns remain relatively stable, with certain transactions occurring frequently between institutions, while others may rarely be queried. This predictable access pattern provides an opportunity to optimize cache management strategies.

Transaction access patterns follow a long-tail distribution [24], where a few high-frequency transactions account for most of the query traffic, and many low-frequency transactions are seldom accessed. The Zipf distribution is widely used to describe the long-tail distribution of data access patterns. The relationship between frequency f and rank r is given by the following formula:

$$f(r) = \frac{C}{r^s} \tag{1}$$

Where $f(r)$ represents the frequency of the r-th ranked item, C is a constant, and s is the exponent of the distribution, controlling the steepness of the curve.

Traditional cache management strategies like LRU and LFU fail to fully utilize cache resources in such cases. Therefore, accurately predicting transaction access frequencies and dynamically adjusting cache strategies becomes key to improving query performance. Transaction queries have strong real-time requirements, especially in high-concurrency scenarios, where query requests are numerous and response time demands are high. Relying solely on off-chain databases for transaction queries may lead to access bottlenecks. Thus, efficient cache management is necessary to reduce database access pressure while ensuring data timeliness and consistency.

In summary, the characteristics of transaction data in consortium blockchains necessitate storage and query optimization strategies that are efficient, dynamically adaptable, and precise.

3.3 Transaction Access Frequency Prediction and Cache Optimization Strategy Based on GCN and XGBoost

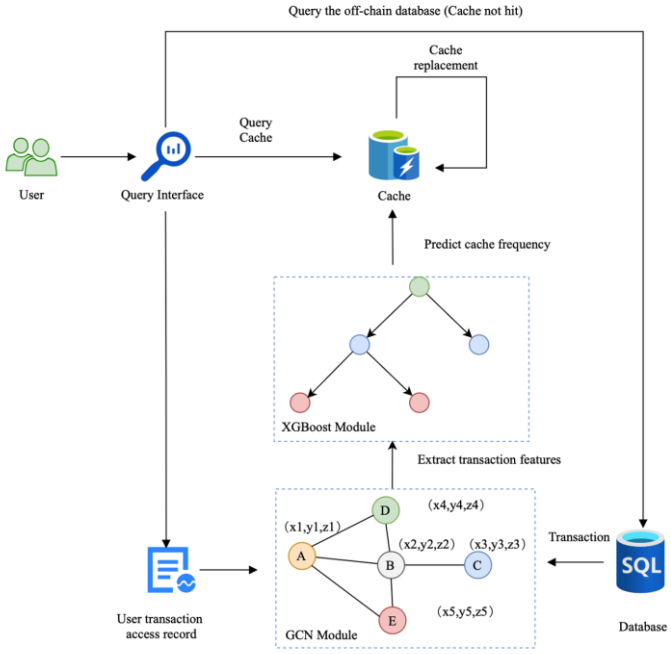

Fig. 2. Query Flowchart

Fig. 2 illustrates the overall workflow for processing user transaction queries. Upon receiving a query via the interface, the system first checks the cache for the requested data. If the cache hit occurs, the result is returned immediately; if the cache miss occurs, the off-chain database is queried and the retrieved result is written back to the cache. Concurrently, the system continuously records users' transaction access histories and, together with the complete transaction data set, constructs a relationship graph within the GCN module to extract node embeddings. These embeddings are concatenated with the original transaction attributes and fed into the XGBoost module to predict each transaction's future access frequency. Based on these predictions, the cache is dynamically updated. High frequency transactions are preferentially retained and low frequency transactions are evicted, thereby substantially improving the cache hit rate, alleviating the database load and reducing query latency.

In this paper, GCN is used to extract the structural features of transaction data. First, we construct a relationship graph of transaction data, where each transaction corresponds to a node in the graph, and the relationships between transactions (such as the same sender or receiver) are represented as edges in the graph. To represent these relationships, we construct the adjacency matrix A, where each element A_{ij} indicates whether there is a transaction relationship between node i and node j. For graph

processing, we normalize the adjacency matrix, obtaining the normalized adjacency matrix $\hat{A}$, defined as:

$$\hat{A} = D^{-1/2} \times A \times D^{-1/2} \tag{2}$$

where D is the degree matrix, with elements $D_{ii} = \sum_j A_{ij}$, representing the degree of node i.

The core of GCN is the graph convolution operation. Graph convolution updates node features by aggregating information from neighboring nodes. The formula is:

$$H^{(l+1)} = \mathrm{ReLU}\left(\hat{A}H^{(l)}W^{(l)}\right) \tag{3}$$

where $H^{(l)}$ is the feature matrix of nodes at the l-th layer, representing the feature representations of all nodes; $\hat{A}$ is the normalized adjacency matrix, indicating relationships between nodes; $W^{(l)}$ is the weight matrix at the l-th layer; and ReLU is the activation function, introducing non-linearity.

The feature matrix $H^{(0)}$ is used to represent all relevant information for each transaction and serves as the input to the Graph Convolutional Network (GCN) model. The initial feature matrix is composed of three parts: sender embeddings, receiver embeddings, and transaction numerical features. The construction process is as follows:

To convert the discrete user IDs into continuous feature vectors, we use an embedding layer. Assuming there are U users and the embedding dimension is d, each user ID is mapped to a d-dimensional vector. Through the embedding layer, the sender and receiver IDs are mapped to the following embedding vectors:

$$\mathbf{e}_i^{(s)} = \mathrm{Emb}(\mathbf{s}_i) \in \mathbb{R}^d \tag{4}$$

$$\mathbf{e}_i^{(r)} = \mathrm{Emb}(\mathbf{r}_i) \in \mathbb{R}^d \tag{5}$$

where $\mathbf{s}_i$ and $\mathbf{r}_i$ are the sender and receiver IDs for the i-th transaction, respectively, and $\mathrm{Emb}(\cdot)$ denotes the embedding operation, outputting a d-dimensional embedding vector. The sender and receiver embeddings are represented by $\mathbf{e}_i^{(s)}$ and $\mathbf{e}_i^{(r)}$, respectively. Other numerical features in the transaction data (such as transaction amount) are normalized. To ensure uniform scaling of the features, we use Min-Max normalization to map the original numerical feature vector $\mathbf{x}_i$ to the range $[0,1]$. The normalized features are:

$$\tilde{\mathbf{x}}_i = \frac{\mathbf{x}_i - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})} \in [0,1]^m \tag{6}$$

where m is the dimensionality of the numerical features, representing the normalized feature matrix $\tilde{\mathbf{x}}_i \in \mathbb{R}^m$.

The final initial feature matrix $H^{(0)}$ is obtained by concatenating the sender and receiver embedding vectors and the normalized numerical features. The initial feature vector $\mathbf{h}_i^{(0)}$ for each transaction consists of the following three parts:

$$\mathbf{h}_i^{(0)} = \left[\mathbf{e}_i^{(s)} \| \mathbf{e}_i^{(r)} \| \tilde{\mathbf{x}}_i\right] \in \mathbb{R}^{2d+m} \tag{7}$$

where $\|$ denotes the concatenation operation, and $2d + m$ is the dimensionality of each transaction's feature vector.
The feature matrix $H^{(0)}$ is then constructed by stacking the feature vectors of all transactions:

$$H^{(0)} = \begin{bmatrix} \mathbf{h}_1^{(0)} \\ \mathbf{h}_2^{(0)} \\ \vdots \\ \mathbf{h}_N^{(0)} \end{bmatrix} \in \mathbb{R}^{N \times (2d+m)} \tag{8}$$

where N is the number of transactions, and $2d + m$ is the length of each feature vector. This matrix $H^{(0)}$ will be passed as input to the Graph Convolutional Network (GCN) for further processing.

To better learn higher-order relationships between nodes, GCN updates node features progressively through multiple layers of convolution. The output of each layer serves as input for the next, enabling the model to learn more abstract feature representations. At the final layer of the GCN model, the feature vector of each node not only contains its own information but also incorporates features from its neighboring nodes, allowing the model to capture more complex relationships between nodes.
The training of GCN is conducted by minimizing a loss function. For classification tasks, we use the cross-entropy loss function, defined as:

$$\mathcal{L} = -\sum_i y_i \log(\hat{y}_i) \tag{9}$$

where y_i is the true label of node i, and $\hat{y}_i$ is the predicted probability by the model.

By modeling the relationship graph of transaction data using GCN, we obtain embedding vectors for each transaction node. These embeddings effectively capture the complex relationships between transaction nodes and provide more expressive features for the subsequent prediction of access frequencies. In this study, the node embeddings learned by GCN, along with other transaction features (such as transaction amount, timestamp, etc.), are input into the XGBoost model to further optimize the prediction of access frequencies.

XGBoost is an ensemble learning method based on decision trees, known for its efficient training speed and strong predictive capability. In this study, XGBoost is used to predict transaction access frequency, which in turn optimizes the cache replacement strategy. The core idea of XGBoost is to build a series of decision trees and modify each tree based on the residuals from the previous round, thereby gradually improving the model's predictive accuracy.

In XGBoost, the goal of model training is to minimize the weighted loss function. Specifically, the loss function includes the loss for each sample as well as a regularization term, as shown in the following formula:

$$\mathcal{L} = \sum_{i=1}^{n} \mathcal{L}_i + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{10}$$

where $\mathcal{L}_i$ is the loss for each sample, typically using mean squared error or log loss; $\Omega(f)$ is the regularization term, which includes the number of trees T, the weight w_j of each leaf node, and regularization parameters γ and λ to control the complexity of the model and prevent overfitting.

The training process of XGBoost involves progressively building decision trees and optimizing the model using gradient boosting methods. In each iteration, XGBoost calculates the error between the current predictions and the true values, and adjusts the model accordingly, gradually reducing the prediction error. This process uses regularization techniques to prevent the model from becoming overly complex, thereby enhancing the model's generalization ability.

In this study, the XGBoost model is used to predict the transaction access frequency based on the features extracted by GCN. First, the node embedding vectors extracted by GCN are combined with the original transaction features to form a complete feature matrix. Then, XGBoost is trained on the combined features to learn the access frequency patterns of the transaction data. The model's performance is optimized by tuning several key hyperparameters, including learning rate, number of trees, maximum depth, and subsample ratio. The learning rate controls the contribution of each tree, with a value of 0.1 chosen based on empirical tests. The number of trees is set to 100, and the maximum depth of each tree is set to 6 to balance model complexity and overfitting.

To address class imbalance, the scale_pos_weight is set to 10, and both subsample and colsample_bytree are set to 0.8 to introduce randomness and prevent overfitting. These hyperparameters are selected through cross-validation and tuned to achieve the best balance between model complexity and prediction accuracy.

After training, the XGBoost model outputs the predicted access frequency for each transaction, which serves as the basis for the intelligent cache replacement strategy. This combined prediction method using GCN and XGBoost can more accurately predict the access frequency of transaction data, optimize cache management strategies, and improve the query performance of the system.

3.4 Cache Replacement Method

In this study, the core of the cache replacement strategy is to use the predicted transaction access frequencies to compute a cache replacement score for each transaction, thereby dynamically determining which data should be prioritized for caching. Specifically, for each transaction t_i, we define the cache replacement score S_{t_i} as follows:

$$S_{t_i} = F_{t_i} \times W_{t_i} \tag{11}$$

where F_{t_i} represents the access frequency after time decay and W_{t_i} denotes the weight corresponding to the transaction category, reflecting the caching priority of different transaction types.

The time-decayed access frequency F_{t_i} is computed using an exponential decay model:

$$F_{t_i} = F_{t_i}^{prev} \times e^{-\lambda \Delta T} + 1 \tag{12}$$

$F_{t_i}^{prev}$ is the previous access frequency of transaction t_i, λ is the time decay factor, and ΔT is the time interval between the current time and the last access time. This model assigns higher weights to recently accessed data while diminishing the influence of older accesses on cache replacement decisions.

The frequency weight W_{t_i} is determined based on the predicted access frequency $F_{t_i}^{pred}$ predicted by XGBoost, and is defined as:

$$W_{t_i} = \omega\left(F_{t_i}^{pred}\right) \tag{13}$$

where $\omega\left(F_{t_i}^{pred}\right)$ is a preset classification function that assigns different weights to different transaction categories based on the predicted access frequency, categorizing transactions into low, medium, and high-frequency groups to determine the cache priority.

Finally, the system makes cache replacement decisions based on the S_{t_i} values. When the cache capacity is reached, the system evicts the transaction with the lowest score and inserts new data, ensuring that high-frequency transactions are preferentially retained in the cache, thereby optimizing cache utilization and reducing query latency.

4 Experiment

4.1 Experiment settings

Experiment Environment.
The experiments were conducted on a machine equipped with an AMD EPYC 7763 CPU, an NVIDIA RTX 4090 GPU, and 384 GB of RAM.

The system runs Ubuntu 18.04.6, and all code was implemented and executed using Python 3.8.19.

Transaction Dataset Construction.
The experimental data consists of synthetic transaction records generated using a Zipf distribution to simulate the long-tail access pattern in consortium blockchain transactions. The dataset includes 10,000 transactions and 1,000 users.

Zipf distribution is used to select a subset of users as high-frequency users, assigning them a higher probability of being chosen as senders and receivers; transaction amounts are generated from a uniform distribution, and timestamps are created at one-minute intervals.

Access frequencies are generated using a Zipf distribution are appropriately increased for transactions involving high-frequency users or high transaction amounts.

Finally, a transaction dataset containing transaction IDs, senders, receivers, transaction amounts, timestamps, and access frequencies is constructed for subsequent experiments on cache optimization and access frequency prediction. By reading the transaction dataset and based on the access frequency of transactions, construct a uniformly distributed transaction access flow.

This synthetic dataset simulates a long-tail access pattern between a small number of high-frequency traders (large carbon emission groups) and many low-frequency traders (small and medium enterprises or individuals), accurately reproducing the characteristics of carbon trading markets where a few large-volume transactions generate frequent queries. At the same time, the uniformly distributed timestamps and access flows align with carbon trading platforms' requirements for historical data auditing and real-time monitoring, making it particularly suitable for evaluating cache optimization and access frequency prediction strategies in carbon trading systems.

Experiment Design.

This experiment consists of three main parts. First, we evaluate the effectiveness of different models in predicting transaction access frequency by comparing our GCN + XGBoost approach against three widely used baselines. XGBoost is included both as the canonical gradient-boosted decision tree and as an ablation baseline to isolate the benefit provided by the GCN-derived embeddings. Notably, prior work [25] has successfully applied XGBoost to classify content popularity for cache updates, demonstrating its effectiveness in cache optimization tasks and further validating its role as a strong baseline in our study. Decision trees have also been used as baselines in the paper, so we additionally employ the common tree-based models LightGBM and Random Forest. LightGBM is selected for its optimized leaf-wise tree growth and histogram-based binning, which deliver dramatically faster training and reduced memory consumption on large transaction logs. Random Forest serves as a robust ensemble of bagged decision trees, chosen for its ability to mitigate overfitting and capture heterogeneous patterns across multiple features without extensive hyperparameter tuning. By benchmarking all four methods on accuracy, F1 score, and recall rate, we clearly quantify how graph-enhanced features improve prediction precision over these classical machine-learning models.

The second part evaluates cache hit rates of different methods. Experiments were conducted with varying cache capacities to compare our proposed method against traditional LFU caching strategy and cache hit rates under access frequency predictions from other machine learning models. These experiments validate the cache optimization effectiveness of the proposed approach.

The final section investigates the impact of different Zipf parameters on cache hit rates, further assessing the adaptability of our method under long-tail distributions. By configuring different Zipf parameters, we observe their effects on caching strategies, thereby evaluating the flexibility and performance of the Proposed method under various distribution scenarios.

4.2 Results and analysis

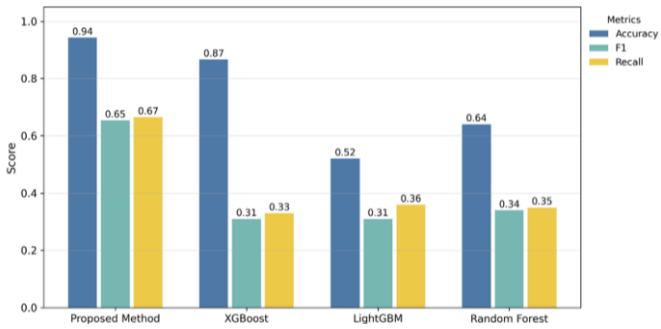

Fig. 3. Comparison of Performance in Predicting Transaction Access Frequency for the Proposed Method and Baseline Strategies

In Fig. 3 regarding the effectiveness of different models in predicting transaction access frequency, the proposed method demonstrates outstanding performance across all evaluation metrics. Compared with other baseline models, the Proposed Method significantly outperforms traditional machine learning models in terms of accuracy, F1-score, and recall. Although XGBoost achieves competitive results in accuracy, it fails to capture potential interaction patterns between transactions like GCN.

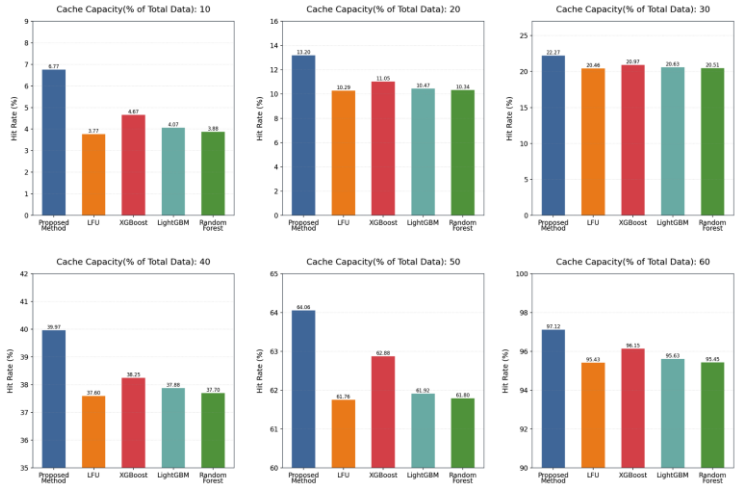

Fig. 4. Comparison of Cache Hit Rates under Varying Cache Sizes for the Proposed Method and Baseline Strategies

As shown in Fig. 4, all methods show improved cache hit rates as cache capacity increases. Under smaller cache capacities, the Proposed Method exhibits notably higher cache hit rates compared to traditional LFU and other machine learning models. As the

cache capacity further expands, the Proposed Method demonstrates significant enhancement in cache hit rate.

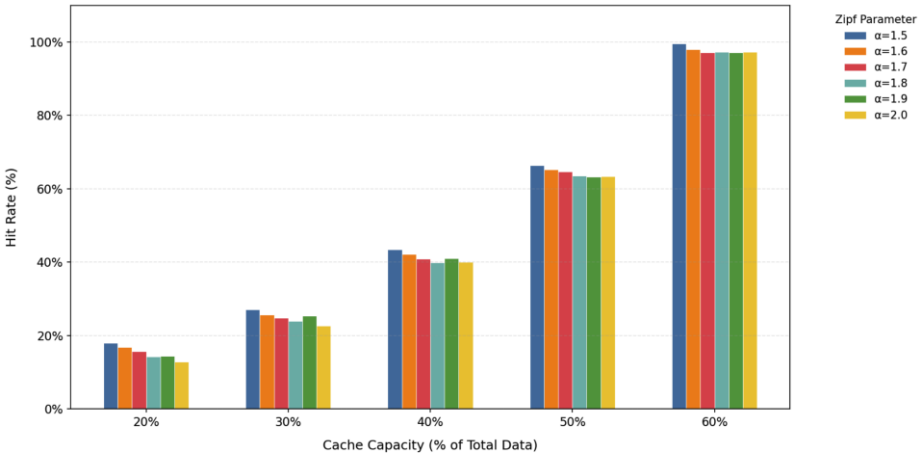

Fig. 5. Impact of Zipf Parameter (α) on Cache Hit Rate for the Proposed Method

Finally, as shown in Fig. 5 when assessing the impact of different Zipf parameters on cache hit rates, the Proposed Method maintains stable adaptability as the Zipf parameter varies. Regardless of changes in the Zipf parameter (α), the Proposed Method sustains high cache hit rates, with gradual improvements observed as cache capacity increases. Particularly under larger Zipf parameters, the Proposed Method achieves optimal cache hit rate performance, indicating robust adaptability to access frequency distributions of varying steepness.

In conclusion, the Proposed Method demonstrates superior effectiveness in transaction access prediction and cache management, particularly excelling in high-concurrency environments by effectively improving cache hit rates while maintaining strong adaptability to diverse access frequency distributions.

5 Conclusion

The architecture proposed in this paper builds upon the traditional on-chain and off-chain systems in blockchain by adding a cache module to optimize query efficiency in high-concurrency environments. The combination of on-chain and off-chain storage in blockchain effectively alleviates data storage pressure, but in high-concurrency situations, traditional cache management methods often fail to meet the demand for fast queries. Therefore, this study introduces an intelligent cache replacement mechanism with GCN and XGBoost models to optimize the cache management strategy, improve the accuracy of transaction data access frequency prediction, and increase cache utilization through smart replacement, further reducing the database access pressure. In this way, the system not only improves query efficiency but also adapts to dynamically

changing transaction access patterns, enhancing overall performance and response speed.

Experimental results show that the proposed method significantly outperforms traditional machine learning models in predicting transaction access frequency and clearly surpasses traditional LFU methods and other machine learning models in cache hit rates. Additionally, the proposed method demonstrates strong adaptability under different Zipf parameters, maintaining stable cache hit rates regardless of the steepness of the access frequency distribution.

In conclusion, the cache optimization method combining GCN and XGBoost offers significant advantages in improving the query performance and response capabilities of blockchain-based transaction systems, especially in high-concurrency and large-scale data processing scenarios. Future research can further optimize the GCN model structure and explore cache management strategies in more complex scenarios.

Acknowledgments. This work was supported by the National Key R&D Program of China(2022YFB2703400)

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. M. Liu et al., "Adaptive shrink and shard architecture design for distributed blockchain," IEEE Transactions on Services Computing, 2019.
2. K. Qian et al., "TDRB: An efficient tamper-proof detection middleware for relational database based on blockchain technology," Information Systems Frontiers, 2021.
3. H. R. Hasan, K. Salah, I. Yaqoob et al., "Trustworthy IoT data streaming using blockchain and IPFS," IEEE Access, vol. 10, pp. 17707–17721, 2022, DOI:10.1109/ACCESS.2022.3146785.
4. G. Al-Sumaidaee, R. Alkhudary, Z. Zilic et al., "An evaluation framework for assessing IPFS performance within a blockchain-based healthcare system," in 2023 IEEE International Conference on Blockchain (Blockchain), 2023, pp. 100–104, DOI: 10.1109/Blockchain60715.2023.00025.
5. S. Chen, X. Tang, H. Wang et al., "Towards scalable and reliable in-memory storage system: A case study with Redis," in 2016 IEEE TrustCom/BigDataSE/ISPA, 2016, pp. 1660–1667.
6. C. Arissabarno, S. Sukaridhoto, and I. Winarno, "Secure & traceable companies file management system using blockchain and IPFS," in 2024 IEEE International Symposium on Consumer Technology (ISCT), 2024, pp. 325–331, DOI: 10.1109/ISCT62336.2024.10791132.
7. A. Maheshwarkar, A. Kumar, and M. Gupta, "Securing database backups using blockchain and peer-to-peer network," in 2021 2nd International Conference on Communication, Computing and Industry 4.0 (C2I4), 2021, pp. 1–6, DOI: 10.1109/C2I454156.2021.9689443.
8. W. Yang, M. Ao, M. Gao et al., "CMSS: A high-performance blockchain storage system with horizontal scaling support," Electronics, vol. 13, no. 10, p. 1854, 2024, DOI: 10.3390/electronics13101854.

9. C. Liu, Y. Li, and Z. Jin, "Blockchain data query scheme based on auxiliary storage," in 2023 5th International Academic Exchange Conference on Science and Technology Innovation (IAECST), 2023, pp. 367–370, DOI: 10.1109/IAECST60924.2023.10503158.
10. E. Zhou et al., "MSTDB: A hybrid storage-empowered scalable semantic blockchain database," IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 8, pp. 8228–8244, 2023, DOI: 10.1109/TKDE.2022.3220522.
11. Y. Rosoon, C. Choksuchat, and P. Aiyarak, "Decentralized trusted database approach to online product reviews," in 2023 15th International Conference on Information Technology and Electrical Engineering (ICITEE), 2023.
12. M. Pandita, D. H. Patil, K. Kashid et al., "Securing blockchain database system: An integrated approach using PCA and isolation forest for intrusion detection," in 2024 5th International Conference for Emerging Technology (INCET), 2024.
13. H. Wang et al., "Optimizing read performance in lightweight blockchain with cooperative storage scheme," in 2023 Fifth International Conference on Blockchain Computing and Applications (BCCA), 2023, pp. 150–157, DOI: 10.1109/BCCA58897.2023.10338901.
14. W. Cai, L. Yu, R. Wang et al., "Blockchain application development techniques," Journal of Software, vol. 28, no. 6, pp. 1474–1487, 2017.
15. Y. Li, K. Zheng, Y. Yan et al., "EtherQL: A query layer for blockchain system," in Proceedings of the International Conference on Database Systems for Advanced Applications, 2017, pp. 556–567.
16. M. El-Hindi, C. Binnig, A. Arasu et al., "BlockchainDB: A shared database on blockchains," Proceedings of the VLDB Endowment, vol. 12, no. 11, pp. 1597–1609, 2019.
17. T. Yu, B. Niu, and X. Fan, "FabricSQL: Querying relational data on blockchain," Computer Engineering and Design, vol. 41, pp. 2988–2995, 2020.
18. K. Miyachi and T. Mackey, "hOCBS: A privacy-preserving blockchain framework for healthcare data leveraging an on-chain and off-chain system design," Information Processing & Management, vol. 58, no. 3, pp. 1–25, 2021, DOI: 10.1016/j.ipm.2021.102523.
19. J. Tu, J. Zhang, S. Chen et al., "An improved retrieval method for multi-transaction mode consortium blockchain," Electronics, 2020.
20. A. Narayanan, S. Verma, E. Ramadan, P. Babaie, and Z.-L. Zhang, "DeepCache: A deep learning based framework for content caching," in Proceedings of the 2018 Workshop on Network Meets AI & ML (NetAI'18), 2018, pp. 48–53, DOI: 10.1145/3229543.3229555.
21. Z. Shi, X. Huang, A. Jain, and C. Lin, "Applying deep learning to the cache replacement problem," in Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '52), 2019, pp. 413–425, DOI: 10.1145/3352460.3358319.
22. J. Hou et al., "A graph neural network approach for caching performance optimization in NDN networks," IEEE Access, vol. 10, pp. 112657–112668, 2022.
23. T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," arXiv preprint arXiv:1609.02907, 2016.
24. L. Breslau, P. Cao, L. Fan et al., "On the implications of Zipf's law for web caching," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 1998.
25. Aziz, Waqar Ali, et al. "Enhancing Edge Caching Efficiency: Leveraging Social and Context-Aware Popularity Rank Prediction via Transfer Learning." 2024 IEEE 30th International Conference on Telecommunications (ICT). IEEE, 2024.