EAI Endorsed Transactions
on Scalable Information Systems

Research Article EAI.EU

Optimization Algorithm for Blockchain Data Storage and Retrieval Based on DHT

Zhijun Xie and Lijuan Zheng*

School of Computer Science, Sichuan University of Commerce, Chengdu 611745, Sichuan, China

Abstract

INTRODUCTION: Each node keeps the identical block data in the decentralized, tamper-proof distributed ledger known as the blockchain.
OBJECTIVES: A blockchain network's working time lengthens, the amount of data that nodes must preserve and synchronize increases noticeably.
METHODS: This brings up significant storage performance difficulties. We have started a study from a blockchain data storage standpoint to address this storage performance issue. We propose a distributed hash table (DHT)-based blockchain data archiving approach by analyzing the redundancy state caused by every node in the current blockchain network containing the same data. The block data is introduced in three ways: archived data building, lookup, and interaction with the underlying chain.
RESULTS: This is done to ensure that blockchain data is not lost and can be accessed. This reduces storage redundancy and satisfies the practical requirements of storage and access in the blockchain's initial application.
CONCLUSION: Experiments on energy transaction data show that the technique suggested in this article has a considerably lower storage occupancy increase rate than fabric storage.

Keywords: Distributed Hash Table; Blockchain; Storage; Archiving; Blocks; Network Nodes

Received on 14 December 2025, accepted on 22 October 2025, published on 02 February 2026

doi: 10.4108/eetsis.8189

*Corresponding Author, Email: zhenglijuan1204@163.com, lijuan_zheng05@outlook.com

1. Introduction

Blockchain technology uses cryptography to assure transmission and access security and is a distributed ledger technology that is maintained by numerous parties [1]. It includes a timestamp-based chained block structure, programmable smart contracts, distributed data storage, distributed node consensus methods, and peer-to-peer transmission networks [2-3]. Consistent data storage, difficulty in tampering, and prohibition of repudiation are characteristics of blockchain [4]. This technological development is becoming the core underpinning technology for many sectors, much like cloud computing, the Internet of Things, and big data technologies [5–6].

The blockchain storage method, to a certain extent, provides the dependability of data and the accessibility and transparency of information, reducing the cost of trust for each system member. Block data will, however, significantly increase as blockchain operation times and the total number of nodes in the network both dramatically increase [7]. The sudden and dramatic growth of data, particularly in high transaction throughput circumstances, may cause access stagnation and system overload.

A collection of keys (key) dispersed among nodes in a distributed system is stored using the Distributed Hash Table (DHT) distributed storage technique [8,9]. This approach can manage circumstances where the nodes' states are not stable and is appropriate for systems with several nodes. The extended network component connects the various nodes in the system so that the nodes can find the data values based on the keywords. The distributed hash table is made up of two key components: the key value space component and the key value component. The key value space component divides the key values into multiple parts and maps them to different nodes in the system.

In terms of storage, each full node in the blockchain must synchronize the data of the entire network, which causes the data of each node to increase over time. In addition, because each node must maintain a copy of the same ledger and because storage pressure is high, it is also impossible to store an excessive amount of data in the blockchain's ledger [10]. In response to the problem that the traditional blockchain requires too much local storage performance for user nodes, This study suggests a blockchain data archiving approach based on DHT to decrease the storage burden on network nodes while maintaining system security, trustworthiness, and decentralization, which, on the premise of not altering the structure and function of the blockchain at all, adopts a storage optimization method based on data archiving, archives block data and distributed storage in a limited number of nodes. Different from the traditional blockchain where each node keeps the same copy of ledger data, this method optimizes the local storage of blockchain nodes, which can reduce the amount of local storage of nodes and lower the storage burden of nodes.

Research Problems: Even though blockchain technology has its upsides like data integrity and security, it still comes across major storage issues in its current systems. When the network increases in size, all full nodes are required to keep the whole record which results in considerable storage redundancy, prolonged data access, and, thus, less scalability. In such cases, off-chain or classic optimization techniques are either too difficult or not enough for federated or multi-domain blockchains.

Research Contributions: The proposed scheme is based on DHT and blockchain for data archiving, which eliminates the problem of storage redundancy by distributing archived blocks only to a certain number of nodes. To achieve this, a B-A tree structure is used for effective block indexing, and a DHT-style index is created for quick lookup, which the optimization of storage allocation and routing of nodes is carried on thus the retrieval speed, scalability and system performance are improved.

2. Related works

The amount of energy data and the number of transactions are expanding exponentially with the continued penetration of the development of the energy industrial Internet[11]. To alleviate the storage pressure brought by the blockchain system, many scholars and experts optimize the block storage scheme in the blockchain. The block storage optimization scheme is mainly to change the block storage method without affecting the transaction throughput [12]. To address the issue of blockchain storage scalability, a method known as the off-chain storage optimization scheme transfers the data content in the block body from the original block body to the off-chain storage system, leaving only the "pointer" and other non-data information pointing to these data to be stored in the block body [13].

The research that has been done by Poovendran and Alagarsundaram has introduced a blockchain-based framework that combines RFID and fog computing to securely collect and share biological signals. This framework guarantees the data's integrity, privacy, and scalability for applications in medical big data. Our proposed method, influenced by this innovative approach, takes on a distributed DHT-based blockchain archiving scheme, where decentralized storage, secure indexing, and efficient retrieval of large-scale blockchain data are the focal points [14].

Literature [15] focuses on the energy transaction process between electric vehicles and distributed networks based on a Byzantine-style blockchain consensus framework. Literature [16] utilizes blockchain technology to manage the system operation of distributed energy transactions. Literature [17] proposed a blockchain energy internet distributed energy trading scheme based on software-defined networks. According to the principle of privacy protection, the approach achieves the reasonable matching of transaction objects. Literature [18] proposed a proof-of-benefit consensus mechanism for peer-to-peer electricity blockchain trading to deal with the current situation of widespread electric vehicle integration. This mechanism enables demand response by offering incentives for balancing local electricity demand in a novel blockchain system. A general framework for a blockchain platform is put forth in literature [19] that supports peer-to-peer energy trading in retail electricity markets, identifies energy matching pairs from both the supply and demand sides, and promotes direct energy trading between producers and consumers, realizing a full energy trading process. A dynamic-reputation practical Byzantine fault tolerance based on dynamic reputation value is proposed in the literature [20]. To increase the transaction security of the slice and efficiently handle cross-slice transactions, literature [21] presents a technique based on jumping hash and dynamic weight slice creation. Literature [22] proposed an algorithm based on jumping hash and dynamic weight slice construction. Literature [23] for PBFT node division stage optimization, node quality improvement in the blockchain system, and failure probability reduction for node integrity issues. The RAFT algorithm is optimized in literature [24] to suit the needs of real-time power data uploading and proposes a distributed energy information interoperability paradigm for the energy field based on blockchain technology. The comparison of popular consensus methods is shown in Table 1.

Table 1. Compared are the common consensus mechanisms

	Strengths	Disadvantages	Purpose
Certificate of Workload	Tolerates attacks up to 51%, fully decentralized	Wasted arithmetic, low token-dependent throughput	First-generation blockchain consensus algorithm
Certificate of Shareholding	Slightly less power consumption, slightly faster block generation speeds	Token-dependent, easy to split, low throughput	Reduce resource consumption and improve efficiency
Delegated Proof of Equity	Not fully decentralized, slightly less power consumption	Token-dependent, low throughput	Solve PoW and PoS resource consumption and efficiency problems
Practical Byzantine Tolerance	Token-independent, slightly less power consumption	Tolerates only 1/3 of malicious nodes, low throughput	Solve the Byzantine problem
Raft	Token-independent, slightly less power consumption	Cannot tolerate malicious nodes, low throughput	Solve node consistency problem

As we can see from the above, the existing consensus mechanisms such as POW and POS consensus mechanisms have the problems of a large number of participating nodes, long consensus time, and low efficiency, and most of them are used in public chains. Similarly, most of the existing blockchain off-chain storage optimization schemes are based on public or private chains, and there are fewer storage optimization schemes in federated chains.

3. DHT-based blockchain data storage

3.1 Design Concept

We suggest a DHT-based blockchain data archiving technique that, in essence, splits a full blockchain into many portions and stores them in the system in a distributed fashion. The fundamental concept is as follows:

Each piece of zone block archived data is tagged by a set of keywords ZB-CID (zone block content identifier). A hash operation is performed on the ZBCID - key = Hash (ZB-CID), which determines the value corresponding to this key, i.e., which network node stores an item in the archive database index.

When looking up the interval block archived data on the chain, the same type of hash algorithm is used for ZBCID to locate the storage node corresponding to this key, and an item in the archived database index, i.e., the archived interval block data, is taken out from the initial node [24-25].

The specific network node value in this design stores a small portion of the interval block data and has no storage pressure itself. When a user needs to obtain the interval block data that is not available on the chain, he or she initiates a request to read the archived interval block data to the system. Through DHT routing, the system obtains locally unstored block data from other nodes with corresponding blocks and finally reconstructs all the block data required by the user.

3.2 Archived data construction

However, because the blockchain will block contained transactions with Merkle tree organization, Merkle tree root for the root of the transaction tree, with the growth of the system running time, the blockchain system in the user more and more, contains more and more nodes account information, the transactions carried out between the account is also increased, the Merkle tree is more and more luxuriant, the number of blocks and so on more and more lead to the number of Merkle tree more and more. The factors mentioned and the number of two dimensions lead to the storage capacity guarantee, that transaction throughput is not optimistic.

For this reason, this paper designs a new attribute structure B-A tree, which not only realizes the function of the Merkle tree but also optimizes the storage and improves the access speed.

The design of node data items based on the B-A tree contains the following data items: a node of a B-A tree includes the B-A tree root B-A (root), the B-A tree mapping node address (K), the B -A tree establishment timestamp, the start block number and end block number of the interval block, and the leaf node left and right pointers (L1, R1).

The data markers during B-A tree establishment are as follows:

(1) A certain range of interval block archived data is noted as$B$, and the original block data of the blockchain system is noted as b;

(2) The height of the tree is denoted as H;

(3) The leaf nodes are denoted as, where n denotes the node from the left to the right.

The B-A tree establishment process is shown in Figure 1 below:

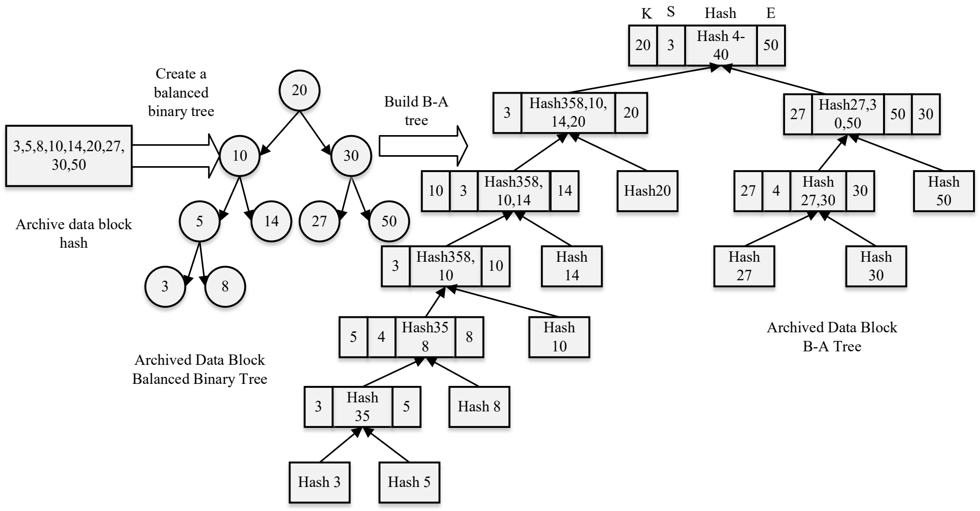


Figure 1. B-A tree construction process

(1) According to the archiving module's configuration conditions interval block size, set the block data included in the written interval block, noted as $B = [b_1, b_2, b_3, \cdots]$;

(2) Take the node account address as the data content of the leaf node, two by two hash merge, that is, equation (1):

$Hash(3) = Hash(1) U Hash(2)$ (1)

Specifically the left leaf node and its parent node hash, i.e., equation (2)

$Hash(tmp\,1) = Hash(\text{ left }) \cup Hash\ (\text{parent})$ (2)

Then the obtained merged hash is merged again with the right leaf node, i.e., equation (3):

$Hash(tmp\,2) = Hash(tmp\,1) \cup Hash(\text{ right })$ (13)

Get the parent node and the left and right leaf nodes merged in a hash, i.e., equation (4)

Hash (root) = Hash $(tmp2)$ (4)

Layer by layer upward recursive merging, the height difference between the left and right two subtrees of each node in the tree during the merging process is satisfied, i.e., equation (5):

$|H_2 - H_1| \leqslant 1$ (5)

In the DHT-based blockchain data archiving scheme suggested by the authors, each block is not replicated all over the network but instead is kept only in a chosen part of nodes. A DHT mechanism that maps and distributes the storage responsibilities based on the node id and hash value is used to find the subset. A fixed replication factor (usually 10-30% of the total nodes) allows for redundancy and fault tolerance but at the same time mainly cuts down the storage and synchronization overhead. During the retrieval process, the DHT lookup quickly finds the required data from the existing replicas and reconstructs it. This method uses storage resources more effectively, allows for more users, and at the same time, ensures the integrity and availability of archived blockchain data if node outages occur.

(3) Repeat the above process over and over again until it is merged into a hash value, so that the storage structure of the B-A tree is built and this root plant is saved in the data itemHash (root).

In the course of creating the archive, a cryptographic hash function is used to generate an identifier for each interval block which is then recorded as a unique index key in the DHT. This key forms a DHT-type mapping connection between the archived block and the nodes that are in charge. The DHT routing table keeps track of these key-node relationships allowing for quick and large-scale data access. The DHT routing system, when a query is made, finds the intended node using the hash key in log N time, which guarantees fast access and reduces the broadcasting costs to a minimum throughout the blockchain network.

3.3 Archived Data Lookup

The data labeling in the archiving process is as follows:

(1) A certain range of interval block archiving data is labeled as$B$, the original block data of the blockchain system is labeled as$b$, the hash value of each block on the chain is labeled as$hash(b_n)$, and the hash value of the archived block is labeled as $hash(B_n)$;

(2) The address on the chain in the node blockchain is noted as BAddr;

(3) The address of the node in the network, i.e., the node value, is denoted as$n(num)$, where num denotes the size of the node value and is of integer type;

(4) The set of node values in the network is$N$, denoted as $N = [n_1, n_2, n_3, \cdots]$;

(5) The ZBCID symbol, which represents the hash value of the archive block in the blockchain and is a hexadecimal string, is used to designate the content identification of the archive block;

(6) The DHT resource identifier of the archive block is noted as ZBKey and is a numeric value less than or equal to $m$bits.

The following describes the method of creating archived data:

(1) The system verifies the stored data, i.e., a certain range of block data in the blockchain, denoted as $B = [b_1, b_2, b_3, \cdots]$;

(2) Calculate the node value: the system uses an m-bit integer as the unique identification of the node, i.e., the node value, and the node identification takes a value in the range of 0 ~ 2^{m-1};

Figuring out each entire node's node value in the DHT archive network. The hash operation is used to extract the node value from the address on the node chain, i.e., the node value of each node is guaranteed to be unique through the hash operation, and here the node value can be regarded as the identity of the node in the DHT archive network.

(3) Construct the identification ring: the node value is sorted, in the clockwise direction in the order of smallest to largest surrounded by a full node identification ring, node$n$ and its closest node for its successor node, denoted as $successor(n)$ the computation process is$N = sort(n1, n2, n3 \ldots)$, $N$is the DHT archiving network sorted set.

For instance, there are 11 nodes with the node values 0, 1, 8, 14, 21, 32, 38, 42, 48, 51, and 56 in a node identification ring with m = 6. The node with ID 1 will take care of the resource with Key 1, the node with ID 14 will take care of the resource with Key 10, and the node with ID 32 will take care of the resource with Key 24, i.e., $successor(1)$ = 1, $successor(10)$ = 14, $successor(24)$ = 32.

(4) Make a routing table: A routing table of length is maintained by each node in the system$m$, storing information about up to$m$ other nodes, and all the nodes are linked together through the routing table. Suppose a node is recognized as$n$, the ith node the first node in its routing table whose value is greater than or equal to$n + 2^{i-1}$, i.e., equation (6):

$$\text{num} = \text{successor}\ (n + 2^{i-1}), 1\text{im} \ (6)$$

The ith node in the routing table of the node$n$ is denoted as$n.finger[i]$.

The node difference value indicates the distance between nodes, and the routing table is established by the node difference operation.

(5) Calculate the archive block hash mapping: hash operation is performed on the archive interval block data requested by the system to obtain the content identifier ZBCID of the archive interval block data, i.e., equation (7):

$$ZBCID = hash(B_n) = hash(b_1 \cup b_2 \cup b_3 \cdots \cup b_n) \ (7)$$

As the keyword; then hash the keyword ZBCID to get the DHT resource identifier of the archived block, i.e.,$ZBKey = hash(ZBCID)$

(6) Archived data lookup:

1) Suppose a node n receives a data lookup request for finding archive block ZBCID, it first hashes its ZBCID to get the archive block DHT resource identifier ZBKey.

2) Compare ZBKey with the value interval table of the node that currently issues the archive data lookup request to see whether it is in the interval of the corresponding node identifier and its successor node, i.e., look for $successor(ZBKey)$ to see whether it is in$(n, n.successor)$, and if it is in the interval, it means that this archive block is in the charge of its successor; otherwise, go to 2);

3) Node n traverses the table entries from front to back

in the routing table until it finds a node whose node value is later than ZBKey, and then passes the ZBKey to this node for it to perform the above lookup, and finally locks to the successor node of ZBKey, i.e., the storage node of the data of the archived block, and establishes communication to complete the data transmission.

Figure 2 shows the process of finding ZBCID = A58790C74696261736520697320617765736F6D6521205C 6 F2, ZBKey = 54 from node 8. The nodes in the routing table of node 8 are 14, 21, 32, and 42, it will first find the first node in the routing table after 54 as 42 from back to front, and then it will request 42 to help find 51. The routing table of node 51 is 56, 61, 1; finally, node 8 finds its successor that is node 56.

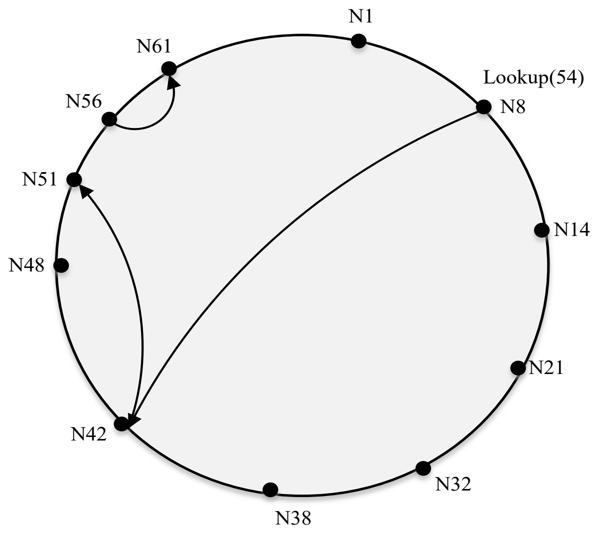


Figure 2. Node lookup archiving block data process

The actual process of constructing a DHT block archiving data system is as follows: assuming that 10 nodes in the blockchain network need to be archived, create a DHT block data archiving network based on this node, and the HASH range of the archiving network is from 1 to 70, and obtain the node value of each node in each network by doing the HASH value of each node's blockchain account address, assuming that the node values obtained by the calculation are 1, 8, 14, 2, 32, 42, 48, 51, 56, and construct a DHT network on a virtual ring with divisions from 1 to 70. values are 1, 8, 14, 2, 32, 42, 48, 51, and 56, to construct a DHT network divided on a virtual ring with HASH interval from 1 to 70, as shown in Figure 3.

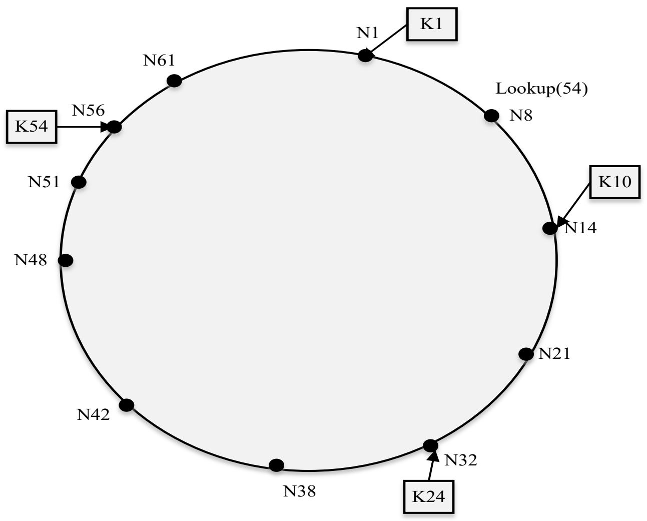


Figure 3. DHT archive network with node value interval from 1 to 70

4 Archiving components interacting with the base chain

3.4 Interaction Structure

For the characteristics of the block structure in different basic chains, the system carries out interface encapsulation and adds an adapter as a bridge module to meet the different archiving storage needs of different basic chains, and different chains need to add different adapter interfaces when interacting with the archiving system, and the design of the interaction structure is shown in Figure 4.

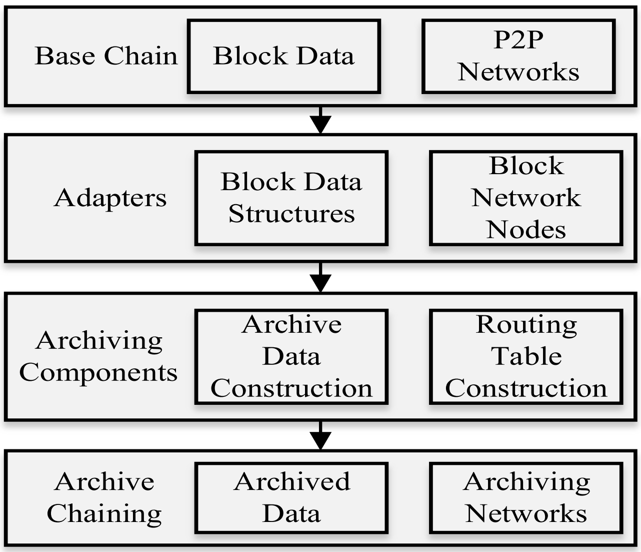


Figure 4. Architecture of chain interaction with DHT archiving component

Among them, the adapter module includes block data structure and blockchain network nodes.

(1) Block data structure: analyze the block structure of different chains and get a unified block structure after normalization, distinguishing blocks into block header and block body.

(2) Blockchain network nodes: the blockchain node address is the account address of the base chain, and each full node stores a DHT archived routing table.

(3) 5 Experimentation and Analysis

For the application of the DHT archiving component to the blockchain system, the following experimental analysis is done from the aspects of transaction throughput and storage compared with the native chain without adding the component to illustrate the superiority of the DHT archiving component.

The consensus module of this storage optimization method is selected using the NCPBFT algorithm [26], and the storage module uses this paper's optimization strategy that was suggested. To offer external services like transaction data and block queries, a local database is used. To maintain system security, the bottom layer of the system continues to use the blockchain technology architectural system, elliptic curve cipher (ECC) asymmetric encryption technique, etc. The simulation experiment is carried out on a server with an Intel Xeon Processor CPU, using OpenStack to virtualize a 17-node ubuntu18.04 system, each node is configured with a CPU frequency of 2.4 GHz, 2 GB of memory, and 20 GB of storage capacity. one monitoring node and four domains are divided into four nodes per domain. The nodes are divided into one monitoring node and four domains, with four nodes in each domain. Hyperledge Fabric 0.6 is chosen to do the comparison experiment in the same environment.

4. Transaction Analysis

The experiment first compares Chain-A, the original blockchain system, and Chain-B, which includes DHT archiving components, in terms of transaction throughput.

The purpose of the experiment is to count the transaction throughput in the two chains of Chain-A and Chain-B, this is determined by how many transactions per second the blockchain system can execute at various numbers of nodes. The results are displayed in Figure 5.

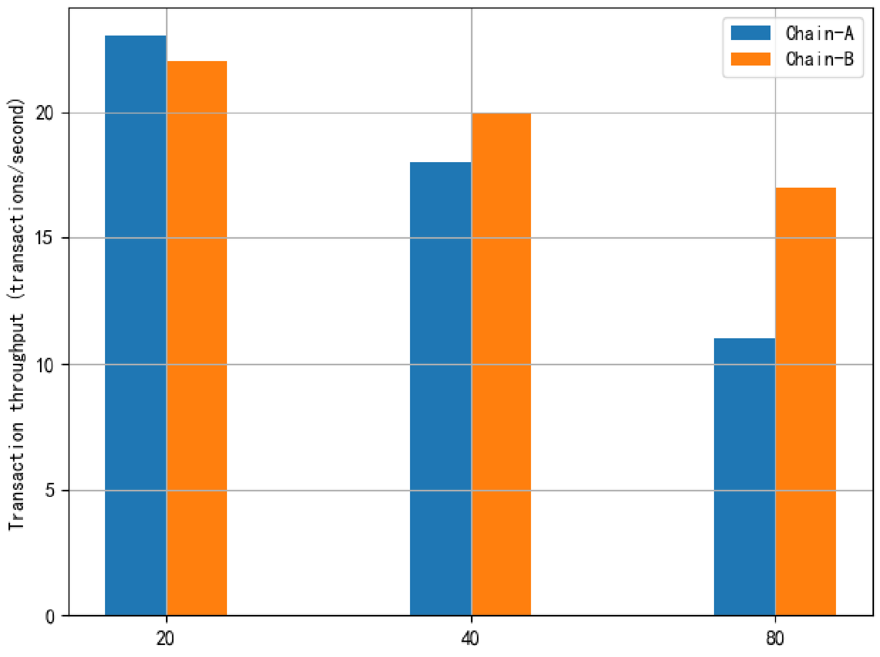


Figure 5. Transaction throughput of Chain-A and Chain-B

When the number of nodes in the network is the same, Chain-A significantly outperforms Chain-B in terms of transaction throughput. The transaction throughput of Chain-B tends to stabilize and is much higher than that of Chain-A as the number of nodes in the network rises. This is because, following the installation of the DHT archiving component, any node in the network may keep data without having to duplicate it throughout the whole network, and the system's overall storage capacity grows smoothly as the number of blocks grows. While each node in Chain-A stores one copy of the network-wide data, as the number of blocks and nodes grows, the double growth factor causes the total network-wide storage to skyrocket, causing transaction blocking, a sharp decline in the number of transactions that can be processed per second, and a sharp decline in transaction throughput.

In addition, the experiment continues to analyze the storage of Chain-A, the native blockchain system, and Chain-B with the DHT archiving component added.

The purpose of the experiment is to count the amount of storage occupied by all nodes in the two chains of Chain-A and Chain-B, and the results are shown in Figure 6.

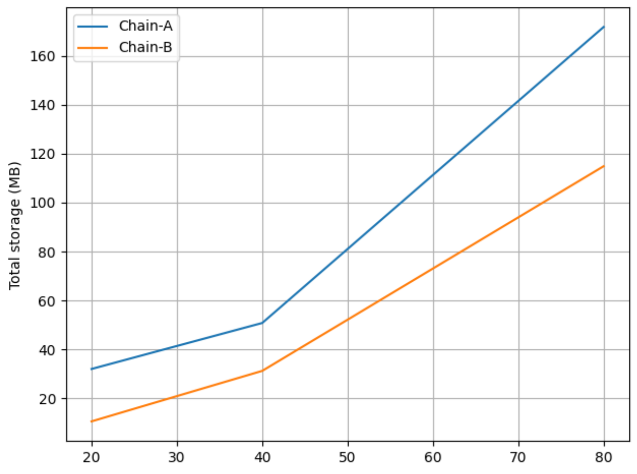


Figure 6 Chain-A and Chain-B storage volume

Because Chain-A's nodes are all full, the difference between the two is relatively small when the number of nodes is low. However, as the number of nodes rises, the occupied storage space Chain-B decreases significantly. When the number of nodes is constant, as the number of blocks rises, the total system storage rises; The overall system storage rises when both the number of nodes and the number of blocks increase; when only the number of nodes or only the number of blocks increases, storage redundancy is more pronounced. In Chain-B, due to the DHT route allocation strategy, as the number of blocks increases, block data is no longer saved in each node, but only in a small number of nodes, and the more nodes there are, the more pronounced the DHT storage advantage becomes, the storage redundancy problem is effectively solved, and the growth of the storage volume of the whole network tends to be flat.

4.1 Node Consensus Performance Test

Due to the difference in the performance of consensus algorithms, to verify the effectiveness of NCPBFT, 4 nodes are tested by sending 1000 energy transaction data and generating 20 consecutive blocks in pairs. As can be seen from Figure 7, compared with several other classical consensus algorithms such as PBFT, PoS-PBFT, RAFT, etc., the average value of the throughput of NCPBFT proposed in this paper has been significantly improved, and the whole transaction processing process takes less time. This is mainly because NCPBFT simplifies the communication complexity of the consensus algorithm and optimizes the master node selection process.

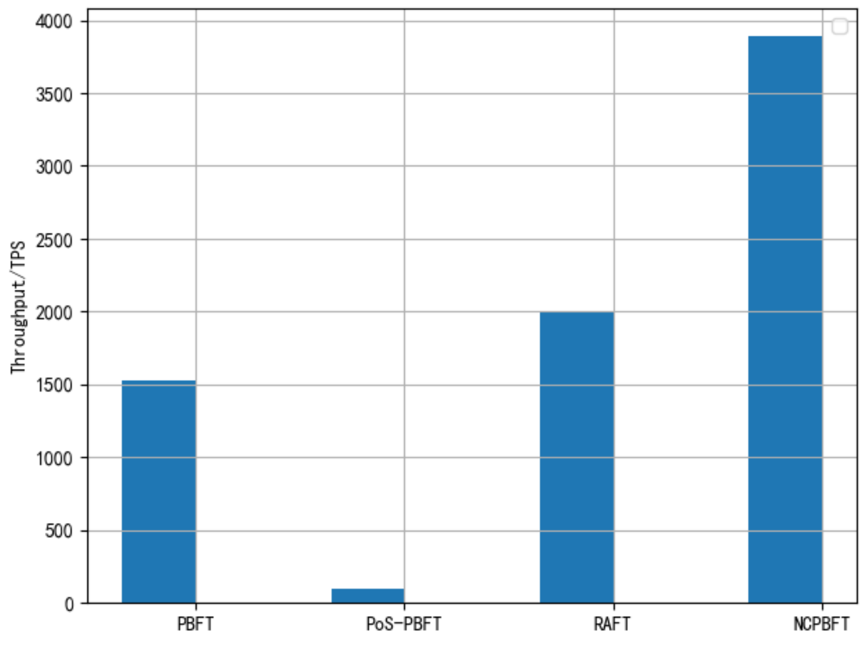


Figure 7. Comparison of throughput of classical consensus algorithms

when there are four nodes involved in the consensus, the block size containing 50, 100, 200, 300, 500, 700, 1000, 1500, 2000, 2500, 3000, and other transaction volumes are selected, and the accompanying experimental analysis is done to compare the NCPDBT method that is suggested in this research with the conventional PBFT algorithm in terms of latency and throughput. Figures 8 and 9 display the precise experimental findings.

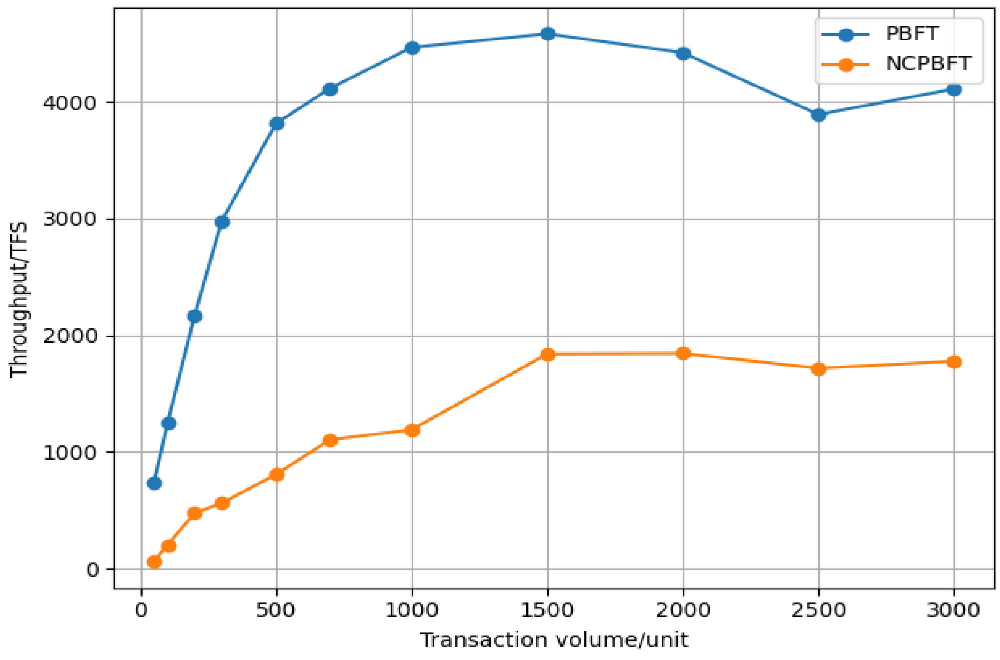


Figure 8. Comparison of throughput in traditional PBFT and NCPBFT with different block sizes

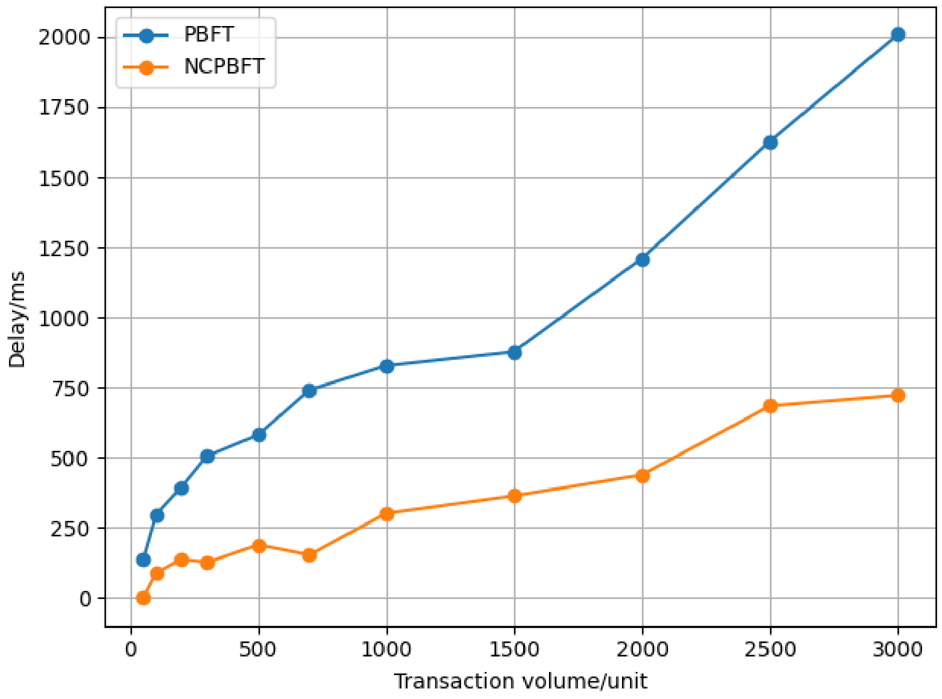
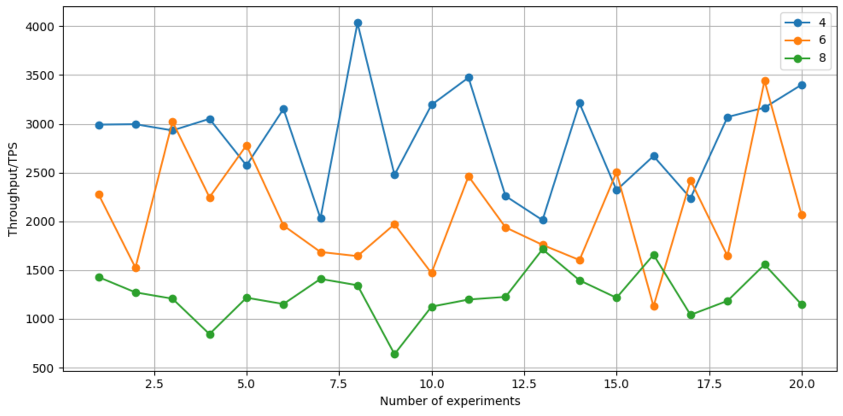


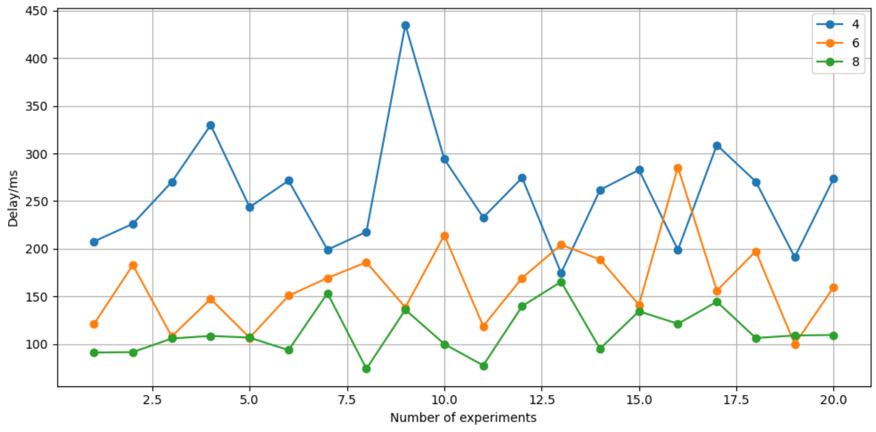
Figure 9. Comparison of traditional PBFT and NCPBFT algorithms in terms of delay when the block size changes.

From Figures 8 and 9, it can be seen that this paper has a higher throughput and lower latency than the PBFT algorithm, the throughput of the two programs at the beginning of the block will increase with the number of transactions and rise, but in the size of the block reaches a certain value after the throughput is slow to grow or even decrease. The latency is increasing with the increase of the block. To achieve the optimum performance, the block size should be set at an appropriate value. Overall, this approach performs better than the PBFT algorithm.

Next, when the block size is 300 and the number of consensus nodes is 4, 6, and 8, the values of throughput and delay changes of the NCPBFT consensus algorithm in 20 experiments are given respectively, the outcomes are displayed in Figure 10.



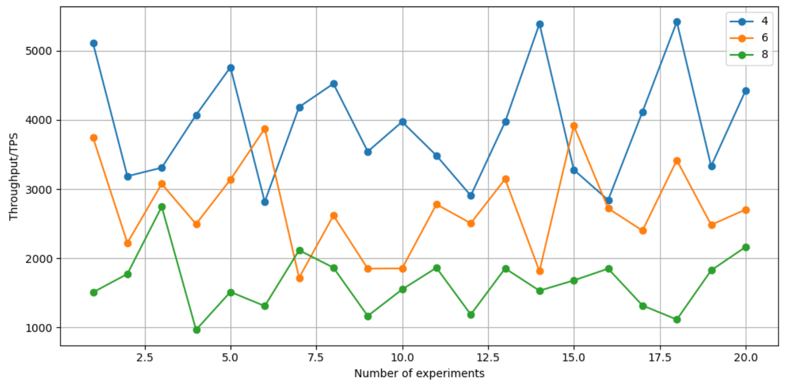
A NCPBFT Throughput



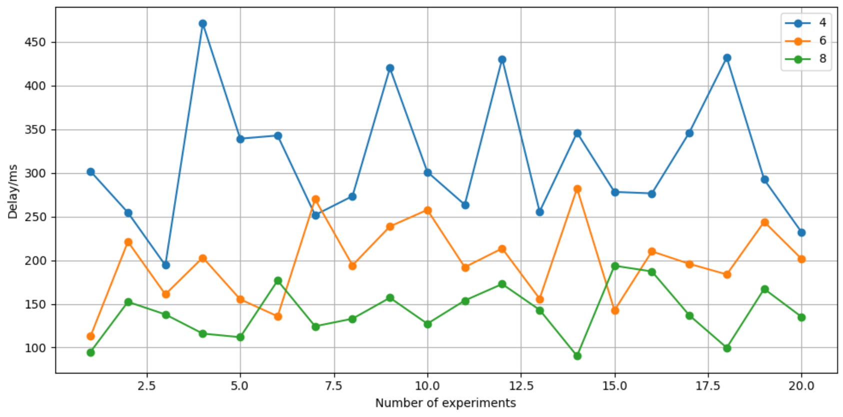
B NCPBFT delay

Figure 10. Performance of the NCPBFT algorithm with different number of consensus nodes when the block size is 300

At block size of 500 and number of consensus nodes are 4, 6, and 8 respectively, the variation values of throughput and delay of the NCPBFT algorithm in 20 experiments are shown in Figure 11.



A NCPBFT Throughput



B NCPBFT delay

Figure 11. Performance of the NCPBFT algorithm with different numbers of consensus nodes when the block size is 500

Figure 10 and 11 show the changes in throughput and latency of the NCPBFT consensus algorithm based on different consensus nodes in several experiments, from which it can be seen that when the block size is fixed, the more consensus nodes are, the lower the throughput will be and the higher the latency will be. This is due to the fact that as there are more nodes, there will be more communications between them, increasing delay and decreasing throughput.

4.2 Storage Overhead Test

When all the nodes in the network work normally and are not attacked, we compare the storage overhead of storing 500, 1000, 2000, and 3000 blocks of data per block under this paper and Fabric storage strategy, and Figure 12 displays the outcomes.

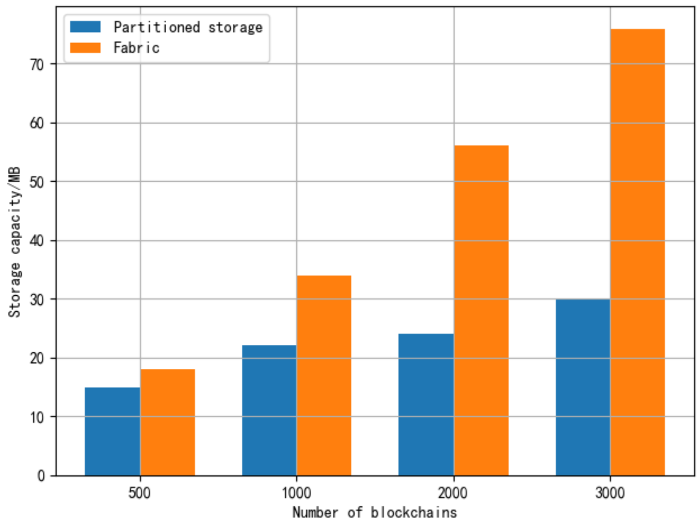


Figure 12. Comparison between the strategy in the paper and the Fabric storage ratio

The following conclusions can be reached from examining the experimental findings in Figure 12.

(1) When there are few blocks generated in the entire blockchain, the storage optimization strategy proposed in this paper is not as obvious as Fabric storage, but when an increase in transaction volume causes a gradual increase in the number of blocks generated in the blockchain, the storage strategy proposed in this paper occupies much less space than Fabric.

(2) This study presents a storage method where the storage occupancy growth rate is substantially lower than the Fabric storage, since an increase in transaction volume causes a slow increase in the block in the blockchain.

Fabric within each node to store the entire block of data, however this work suggests a storage scheme in each node domain where only storage nodes will store the entire block of data, with regular storage nodes just storing the block headers for each block in the block chains; when the number of blocks gradually increase, this paper's advantage of the program is more obvious, compared with the storage occupancy of the Fabric, although also gradually increasing, but its growth rate is much lower than that of Fabric storage. Compared with Fabric, although the storage consumption is also gradually increasing, the growth rate slows down significantly. The experimental findings demonstrate the efficacy of the plan outlined in this study.

4.3 Transaction verification test

The speed of the blockchain system's response to the verification request during the transaction verification process is critical for determining if the transaction verification can produce accurate feedback results, and this verification is mainly based on the processing time of different nodes to the same transaction verification request as a benchmark for verification testing. Assuming that the nodes in the blockchain system are in normal operation, select the storage nodes, ordinary storage nodes, and nodes in Fabric, and compare the verification time required by the same node in querying and verifying 500, 1,000, 2,000, and 3,000 transactions. Figure 13 displays the specific results.

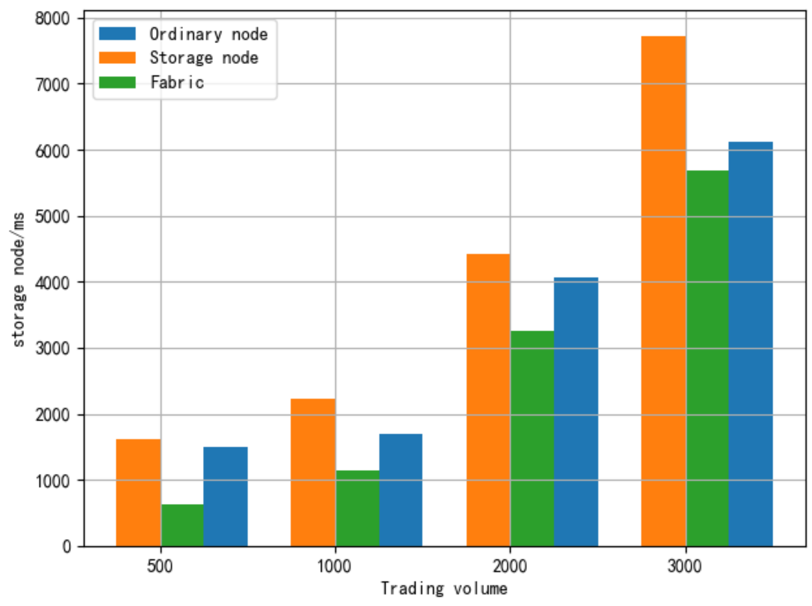


Figure 13. Comparison of transaction validation

The results in Figure 13 allow for the following deductions.

(1) As data volume increases with a certain number of transactions queried, the query validation time on regular storage nodes, storage nodes, and fabric nodes all increases [27].

(2) With the same number of queried transactions, the query verification time is almost the same on the storage node and on the Fabric node [28].

(3) In the case of the same number of query transactions, the time for query validation on ordinary storage nodes is much higher than that on storage nodes and on Fabric nodes.

Under the storage optimization strategy based on partitioning, block data is stored by storage nodes and ordinary storage nodes respectively, in which only the storage nodes have the right to store complete block information, and ordinary storage nodes are only responsible for storing block header information. The verification request initiated by ordinary nodes has to be further forwarded to the storage node to verify the transaction, so the query will be a little slower. all nodes of Fabric store the complete block data, so the query verification time of Fabric nodes is not much different from the query verification time of the storage node. In summary, although the query verification time of common storage nodes is a little longer, it has no impact on how the entire blockchain energy trading system functions normally.

5. Conclusion

The full node in the blockchain saves all the block time from the beginning to the current in the whole network, which causes storage redundancy, high requirements on node storage performance, and long time-consuming synchronization of block data after new nodes join. For this reason, this paper proposes a DHT-based blockchain data archiving scheme, where individual archived blocks are saved in a certain percentage of nodes instead of all nodes. The scheme, when archiving block data in the blockchain, is organized and constructed in accordance with a certain data structure, and the archived block data identifiers are stored using the hash operation to establish a DHT-style index relationship with the nodes and queried through the DHT routing table. After that, the block data allocation strategy between archive zones is optimized to make node storage stress-free and balanced with each other, and the DHT-based routing algorithm is optimized to simplify the path and improve the routing efficiency.

Declarations

Funding

There is no specific funding to support this research.

Acknowledgements

The authors would like to show sincere thanks to those techniques who have contributed to this research.

Consent for publication

All authors reviewed the results, approved the final version of the manuscript and agreed to publish it.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

Data Availability

The experimental data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declared that they have no conflicts of interest regarding this work.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

Author's contributions

All Author is contributed to the design and methodology of this study, the assessment of the outcomes and the writing of the manuscript.

References

[1] Alizadeh, M., Andersson, K., & Schelén, O. (2022). DHT-and blockchain-based smart identification for video conferencing. *Blockchain: Research and Applications*, *3*(2), 100066. https://doi.org/10.1016/j.blcra.2022.100066

[2] Mubashar, A., Asghar, K., Javed, A. R., Rizwan, M., Srivastava, G., Gadekallu, T. R., ... & Shabbir, M. (2022). Storage and proximity management for centralized personal health records using an IPFS-based optimization algorithm. *Journal of Circuits, Systems and Computers*, *31*(01), 2250010. https://doi.org/10.1142/S0218126622500105

[3] Nartey, C., Tchao, E. T., Gadze, J. D., Yeboah-Akowuah, B., Nunoo-Mensah, H., Welte, D., & Sikora, A. (2022). Blockchain-IoT peer device storage optimization using an advanced time-variant multi-objective particle swarm optimization algorithm. *EURASIP Journal on Wireless Communications and Networking*, *2022*(1), 1-27. https://doi.org/10.1186/s13638-022-02122-9

[4] Zhao, J., Zhang, D., Liu, W., Qiu, X., & Brusic, V. (2022). DHT-based blockchain dual-sharding storage extension mechanism. *Applied Sciences*, *12*(19), 9635. https://doi.org/10.3390/app12199635

[5] Nguyen, D. C., Pathirana, P. N., Ding, M., & Seneviratne, A. (2021). BEdgeHealth: A decentralized architecture for edge-based IoMT networks using blockchain. *IEEE Internet of Things Journal*, *8*(14), 11743-11757. https://doi.org/10.1109/JIOT.2021.3069520

[6] Hassanzadeh-Nazarabadi, Y., Küpçü, A., & Ozkasap, O. (2019). Decentralized utility-and locality-aware replication for heterogeneous DHT-based P2P cloud storage systems. *IEEE Transactions on Parallel and Distributed Systems*, *31*(5), 1183-1193. https://doi.org/10.1109/TPDS.2019.2919733

[7] Aslam, S., Bukovszki, V., & Mrissa, M. (2021). Decentralized data management privacy-aware framework for positive energy districts. *Energies*, *14*(21), 7018. https://doi.org/10.3390/en14217018

[8] Liu, T., Wu, J., Li, J., Li, J., & Zhang, Z. (2022). Efficient algorithms for storage load balancing of outsourced data in blockchain network. *The Computer Journal*, *65*(6), 1512-1526. https://doi.org/10.1093/comjnl/bxz128

[9] Zhang, J., Zhong, S., Wang, J., Yu, X., & Alfarraj, O. (2021). A storage optimization scheme for blockchain transaction databases. *Computing and Systems in Science and Engineering*, *36*(3), 521-535. https://doi.org/10.32604/csse.2021.015434

[10] Zhao, Y., Li, Q., Yi, W., & Xiong, H. (2023). Agricultural IoT data storage optimization and information security method based on blockchain. *Agriculture*, *13*(2), 274. https://doi.org/10.3390/agriculture13020274

[11] Hassanzadeh-Nazarabadi, Y., Küpçü, A., & Özkasap, Ö. (2021). LightChain: Scalable DHT-based blockchain. *IEEE Transactions on Parallel and Distributed Systems*, *32*(10), 2582-2593. https://doi.org/10.1109/TPDS.2020.3014869

[12] Feng, H., Wang, J., & Li, Y. (2022). A blockchain storage architecture based on information-centric networking. *Electronics*, *11*(17), 2661. https://doi.org/10.3390/electronics11172661

[13] Fan, X., Niu, B., & Liu, Z. (2022). Scalable blockchain storage systems: Research progress and models. *Computing*, *104*(6), 1497-1524. https://doi.org/10.1007/s00607-022-01008-z

[14] Alagarsundaram, P., & Carolina, N. (2021). Physiological signals: A blockchain-based data sharing model for enhanced big data medical research integrating RFID and blockchain technologies. *J Comput Sci*, *9*(2), 12-32.

[15] Guo, J., Li, C., & Luo, Y. (2022). Blockchain-assisted caching optimization and data storage methods in edge environment. *The Journal of Supercomputing*, *78*(16), 18225-18257. https://doi.org/10.1007/s11227-022-04692-0

[16] Wang, X., Liu, J., & Zhang, C. (2023). Network intrusion detection based on multi-domain data and ensemble-bidirectional LSTM. *EURASIP Journal on Information Security*, *2023*(1), 5. https://doi.org/10.1186/s13635-023-00159-5

[17] Gai, K., Guo, J., Zhu, L., & Yu, S. (2020). Blockchain meets cloud computing: A survey. *IEEE Communications Surveys & Tutorials*, *22*(3), 2009-2030. https://doi.org/10.1109/COMST.2020.2970749

[18] Yang, J., Jia, W., Gao, Z., Guo, Z., Zhou, Y., & Pan, Z. (2023). Cuckoo-Store engine: A Reed–Solomon code-

based ledger storage optimization scheme for blockchain-enabled IoT. *Electronics*, *12*(15), 3328. https://doi.org/10.3390/electronics12153328

[19] Antwi, R., Gadze, J. D., Tchao, E. T., Sikora, A., Nunoo-Mensah, H., Agbemenu, A. S., ... & Keelson, E. (2022). A survey on network optimization techniques for blockchain systems. *Algorithms*, *15*(6), 193. https://doi.org/10.3390/a15060193

[20] Li, D., Luo, Z., & Cao, B. (2022). Blockchain-based federated learning methodologies in smart environments. *Cluster Computing*, *25*(4), 2585-2599. https://doi.org/10.1007/s10586-022-03688-9

[21] Zaabar, B., Cheikhrouhou, O., Jamil, F., Ammi, M., & Abid, M. (2021). HealthBlock: A secure blockchain-based healthcare data management system. *Computer Networks*, *200*, 108500. https://doi.org/10.1016/j.comnet.2021.108500

[22] Mughal, M. H., Shaikh, Z. A., Ali, K., Ali, S., & Hassan, S. (2022). IPFS and blockchain-based reliability and availability improvement for integrated rivers' streamflow data. *IEEE Access*, *10*, 61101-61123. https://doi.org/10.1109/ACCESS.2022.3182704

[23] Putz, B., Dietz, M., Empl, P., & Pernul, G. (2021). Ethertwin: Blockchain-based secure digital twin information management. *Information Processing & Management*, *58*(1), 102425. https://doi.org/10.1016/j.ipm.2020.102425

[24] Praveen, M. D., Totad, S. G., Rashinkar, M., Ostwal, R., Patil, S., & Hadapad, P. M. (2022). Scalable blockchain architecture using off-chain IPFS for marks card validation. *Procedia Computer Science*, *215*, 370-379. https://doi.org/10.1016/j.procs.2022.01.060

[25] Deshpande, V., Badis, H., & George, L. (2022). Efficient topology control of blockchain peer-to-peer network based on SDN paradigm. *Peer-to-Peer Networking and Applications*, *15*(1), 267-289. https://doi.org/10.1007/s12083-021-00386-z

[26] Miyachi, K., & Mackey, T. K. (2021). hOCBS: A privacy-preserving blockchain framework for healthcare data leveraging an on-chain and off-chain system design. *Information Processing & Management*, *58*(3), 102535. https://doi.org/10.1016/j.ipm.2020.102535

[27] Wei, Q., Li, B., Chang, W., Jia, Z., Shen, Z., & Shao, Z. (2022). A survey of blockchain data management systems. *ACM Transactions on Embedded Computing Systems (TECS)*, *21*(3), 1-28. https://doi.org/10.1145/3523071

[28] Ghazali, R., & Down, D. G. (2025). Smart Data Prefetching Using KNN to Improve Hadoop Performance. *EAI Endorsed Transactions on Scalable Information Systems*, *12*(3).https://doi.org/10.4108/eetsis.9110