Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.080210

ARTICLE

An Improved Blockchain-Empowered Storage Service Based on Data Association

Bin Fang[1], Qi Yu[1], Han Wu[2], Xingxing Hou[2] and Jingyu Zhang[2,*]

[1]State Grid Information & Communication Company of Hunan Electric Power Corporation, Hunan Provincial Key Laboratory of Internet of Things in Electricity, Changsha, China

[2]School of Computer Science and Technology, Changsha University of Science and Technology, Changsha, China

*Corresponding Author: Jingyu Zhang. Email: zhangzhang@csust.edu.cn

Received: 04 February 2026; Accepted: 28 April 2026; Published: 15 June 2026

ABSTRACT: The integration of Internet of Things (IoT) with blockchain technology introduces significant challenges in handling massive and frequent transaction data generated by distributed IoT devices. The Unspent Transaction Output (UTXO) model, widely adopted in blockchain systems like Bitcoin, faces critical scalability issues when applied to IoT environments. This is because the datasets it processes expand rapidly, which consumes a large amount of memory and increases the disk access latency of resource-constrained IoT nodes. Existing optimization approaches exhibit limitations in dynamic adaptability and protocol compatibility. To address these challenges, we propose an improved blockchain-empowered storage service that introduces a data association-based multi-zone storage framework for UTXO-based blockchain systems. This service establishes a predictive model for UTXO access probability by analyzing IoT device transaction time intervals and address-specific frequencies. Subsequently, we design a dynamic three-zone storage framework that intelligently optimizes UTXO placement. Experimental results demonstrate that our approach significantly reduces disk access and optimizes memory usage compared with prevalent algorithms, while maintaining compatibility with existing UTXO-based protocols.

KEYWORDS: Blockchain systems; data association; storage optimization; memory utilization; storage services

1 Introduction

The integration of blockchain technology with Internet of Things (IoT) systems has introduced unprecedented scalability challenges, especially in transaction management across decentralized networks. Since the advent of Bitcoin in 2008, blockchain technology has undergone a significant evolution. Its inherent characteristics—decentralization, traceability, and privacy preservation—have enabled its adoption beyond cryptocurrency into diverse industries [1–3]. Fig. 1 illustrates the integrated architectural framework of blockchain-enabled IoT systems and the functional interaction within the Unspent Transaction Output (UTXO) model engine. Nevertheless, scaling blockchain systems compounds the critical challenges for full nodes in IoT environments, as the exponential expansion of the UTXO set imposes profound storage overhead and verification delays.

In UTXO-based blockchain systems like Bitcoin, full nodes must download and store the entire blockchain datasets to independently verify transactions [4]. The surge in UTXO data volume leads to high memory usage and increased disk access latency. This issue directly impacts the scalability and real-time performance required in IoT environments. Although existing optimization solutions have progressed in areas

like storage compression, caching strategies, and sharding technology, challenges remain, such as insufficient dynamic adaptability, protocol compatibility limitations, and the complexity of predictive models.

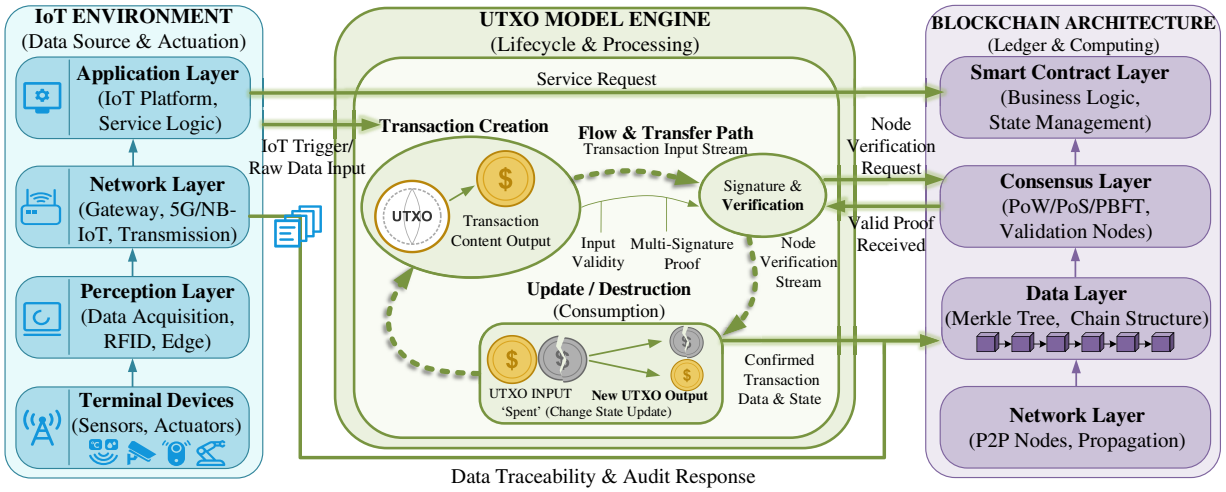

Figure 1: The overall architectural framework of UTXO-based blockchain systems in IoT environments.

To address these challenges, this paper proposes an improved storage service based on data association for large-scale blockchain systems. By establishing data associations between address-related transaction intervals and frequency features of IoT device transactions, a probability-driven dynamic partition UTXO storage framework is developed. Specifically, the core contributions of this research, which distinguish our work from traditional rule-based cache replacement and static partitioning schemes, are as follows:

(1) We establish a data association-driven probabilistic access prediction model for UTXOs, quantify address-related transaction interval correlations and transaction frequency features for active, pre-hoc access probability prediction.
(2) A hot-warm-cold storage service framework is proposed to dynamically store high-probability UTXOs in memory and migrate others to disk, optimizing the performance-storage trade-off.
(3) Extensive comparative experiments are conducted for evaluation, and our approach achieves fewer disk access, shorter block verification latency and lower memory usage compared with prevalent rule-based algorithms.

The remaining structure of this paper is as follows: Section 2 reviews research on UTXO processing and verification optimization. Section 3 discusses UTXO models, common cache replacement algorithms and transaction processing. Section 4 presents the multi-zone dynamic storage service framework and analyzes its performance. Section 5 evaluates the proposed service through simulation experiments. Finally, the last section summarizes and outlines future research directions.

2 Related Work

As the first and most representative blockchain system adopting the UTXO model, Bitcoin faces significant storage and verification challenges in full-node operations—particularly in IoT environments—due to the exponential growth of UTXO datasets. Nakamoto's original design ensures security through chain-based verification but does not address storage efficiency [5]. Over the past decades, substantial efforts have been devoted to optimizing UTXO data processing and verification efficiency.

To alleviate storage pressure caused by dataset expansion, various scaling mechanisms have been proposed. Segregated Witness (SegWit) [6] reduces transaction size by separating signature data, while enlarging block size [7] enhances throughput at the cost of increased storage burden and centralization risks. Off-chain solutions such as the Lightning Network [8] reduce main-chain load via payment channels, and sidechain technologies [9] improve interoperability but introduce additional trust assumptions.

In terms of UTXO dataset compression and storage optimization, Song and Shudo [10] proposed periodic UTXO aggregation to reduce storage occupancy. Nguyen et al. [11] optimized UTXO selection at the transaction level to shrink dataset size. Zhang et al. [12] introduced expiration-based partitioning with LRU (Least Recently Used) mechanisms to migrate cold data. Other representative works include Txilm [13], which compresses transaction size through ordering strategies; Split-scale [14], which increases throughput via tree-based blockchain division; Bitcoin-NG [15], which decouples leader election from transaction serialization to improve processing efficiency; Section-blockchain [16], which effectively reduces large-scale storage requirements; the coloring index structure proposed by Feng et al. [17]; zero-knowledge-based storage reduction schemes [18]; and snapshot-guided synchronization approaches such as CoinPrune [19]. Although these methods reduce redundancy or improve throughput, many rely on protocol modifications or static thresholds that lack adaptability under dynamic IoT workloads.

Transaction verification efficiency is fundamentally constrained by UTXO access latency and cryptographic overhead. Empirical studies demonstrate a substantial performance gap: when UTXO datasets are fully memory-resident, throughput can reach up to 10,000 transactions per second (TPS), whereas disk-based access reduces it to approximately 100 TPS [20]. To mitigate I/O bottlenecks, memory-centric optimization strategies have been explored. Xu et al. [21] designed a cross-block aggregation index to reduce disk operations and improve verification speed. The EBV mechanism [22] streamlines UTXO state representation but requires additional proof submissions. Wang et al. [23] integrated zero-knowledge proofs with Merkle root hashing to reduce block transmission overhead through cryptographic enhancement. Lightweight verification approaches further improve efficiency: Utreexo [24] employs hash accumulators to significantly reduce storage requirements for light nodes, while FlyClient [25] enables near-complete chain verification using only a small fraction of block headers. However, privacy-preserving and cross-chain mechanisms may introduce additional latency and confirmation overhead.

Beyond classical storage and verification optimization, recent work introduces more advanced hierarchical and classified storage mechanisms to mitigate scalability challenges in blockchain-enabled IoT systems. Hierarchical classified storage schemes based on erasure coding significantly reduce storage redundancy while maintaining high query performance and decentralization in smart IoT environments [26]. Meanwhile, joint clustering and storage optimization strategies further improve distributed storage efficiency and minimize access costs in large-scale IoT deployments [27]. In addition, recent blockchain-based IoT storage frameworks incorporate secure data management and hybrid on-chain/off-chain collaboration mechanisms to improve data integrity and reduce storage overhead in distributed environments [28,29].

In parallel, energy efficiency and architectural scalability have become critical research directions. Recent frameworks increasingly leverage Layer-2 scaling, off-chain mechanisms, and edge computing paradigms to reduce energy consumption and latency in IoT-blockchain systems [30]. Studies analyzing energy consumption profiles indicate that disk I/O operations and consensus overhead constitute dominant power costs for resource-constrained devices, motivating more adaptive state management strategies [31]. Additionally, lightweight consensus mechanisms such as Delegated Proof-of-Stake (DPoS), combined with distributed off-chain storage solutions (e.g., IPFS), demonstrate improved throughput and resource utilization compared with traditional architectures [32].

Nevertheless, existing approaches predominantly focus on macro-level architectural redesign, consensus modification, or static storage partitioning. Most existing UTXO storage optimization schemes rely on passive, single-dimensional rule-based mechanisms (e.g., LRU and LFU) that only adjust data placement after an access event occurs, failing to capture the intrinsic correlation between transaction behaviors and future access probability. Fine-grained, probability-driven lifecycle management of UTXO datasets within full nodes, particularly under dynamic and heterogeneous IoT workloads, remains insufficiently explored.

To address these limitations, this paper proposes a data association-based multi-zone storage service (DASS) for UTXO-based blockchain systems in IoT scenarios. By integrating transaction interval and frequency characteristics to construct a probabilistic access model, DASS dynamically partitions UTXOs into hot, warm, and cold zones. Compared with traditional strategies, the proposed approach reduces disk access and optimizes memory utilization without requiring protocol modifications or complex cryptographic restructuring, thereby enabling scalable, responsive, and energy-efficient blockchain operations under variable IoT workloads.

3 Background Knowledge

3.1 UTXO and UTXO Sets

The UTXO model, fundamental to transaction-based blockchain systems like Bitcoin, operates distinctively from account-based models. A UTXO is an abstract ownership of a cryptocurrency [33], which represents a chain of payment relationships between transactions [34]. In this framework, transactions consist of inputs (existing UTXOs) and outputs (new generated UTXOs), where inputs are consumed to generate outputs for payees. Unlike account-based systems that track balances directly, UTXO-based systems aggregate all unspent outputs linked to a user's address to calculate balances, enhancing privacy by preventing account tracking and mitigating double-spending risks. In this paper, all our designs are based on the UTXO-based blockchain systems. However, the growing UTXO sets pose storage challenges: Full nodes must store the entire dataset in memory for efficient verification, but exponential growth forces partial storage on slower disks. This introduces significant access latency, as disk-stored UTXOs require longer verification time than those in memory, highlighting critical trade-offs between scalability and performance.

3.2 Cache Replacement Algorithms

A cache replacement algorithm manages cache space by determining which data should be replaced when new data needs to be loaded. Common algorithms include LRU, MRU (Most Recently Used), LFU (Least Frequency Used), and MFU (Most Frequently Used).

The LRU algorithm eliminates data that hasn't been used for the longest time. It tracks usage through an access order queue, removing data from the queue's end when space is needed. LRU is effective in predicting future access patterns based on recent history, making it simple and efficient for cache space utilization. In contrast, the MRU removes the most recently accessed data. It moves accessed data to the front and eliminates data from the head when necessary. This approach is based on the assumption that recently used data is less likely to be used again. The LFU removes data with the lowest access frequency, assuming that less frequently used data is less likely to be needed again. It assigns a frequency counter to track usage. It is useful in database storage but more complex, as it requires additional structures and decision-making when frequencies match. Similarly, the MFU eliminates the most frequently accessed data, based on the assumption that such data may have diminished value in the cache. It is useful when predicting that frequently accessed data may no longer be needed.

3.3 Transaction Processing Flow

In blockchain systems tailored for Internet of Things (IoT) applications, transaction processing is crucial to ensuring system security and maintaining blockchain integrity. It verifies the legitimacy of each transaction through a series of strict rules and checks, ensuring the blockchain system's decentralized, transparent, and immutable characteristics. As shown in Fig. 2, transaction processing in IoT-supported blockchains consists of six stages: creation, broadcasting, verification, pooling, packaging, and confirmation.

(1) Transaction Creation: An IoT device or gateway generates a transaction using a lightweight cryptographic wallet. The user creates a transaction using their digital wallet application and specifying the payee, amount, and transaction fee. The transaction is signed with the user's private key to prove ownership of the UTXO.

(2) Transaction Broadcasting: Once signed, the transaction is broadcast to the nodes in the blockchain system.

(3) Transaction Verification: Nodes verify the transaction by checking its version, input validity, signature, and timestamp. In IoT environments, this step often emphasizes efficiency to accommodate high transaction volumes from distributed devices.

(4) Transaction Pooling: Verified transactions are placed into the transaction pool, awaiting inclusion by miners in a new block.

(5) Transaction Packaging and Blockchain Addition: Miners select transactions from the pool and bundle them into a new block. They then perform the proof-of-work calculation, and once successful, the new block is added to the blockchain and broadcast to the network.

(6) Block Verification and Confirmation: Nodes verify the validity of the new block and its transactions. Once verified, the block is broadcast. A transaction is considered confirmed when it is included in a block and added to the blockchain.

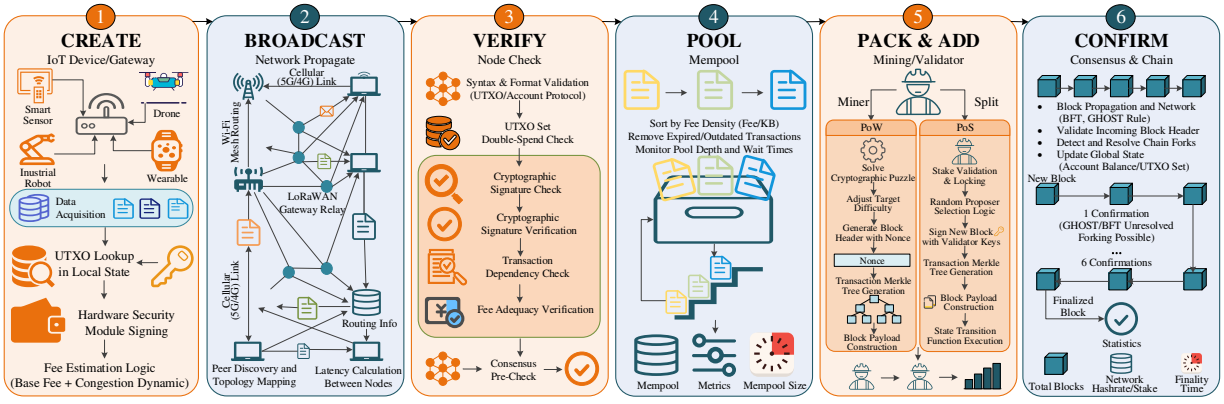

Figure 2: Transaction processing flowchart.

4 Service System Design

4.1 Design of the Multi-Zone Service Framework

The workflow of the multi-zone service framework for managing blockchain transaction datasets in IoT environments is shown in Fig. 3. First, an IoT device generates a transaction and broadcasts it to the network nodes. After verification, the miner broadcasts it to neighboring nodes, then packages the transaction into a block and uploads it to the chain. The proposed framework identifies the address access probability of each transaction after block verification and then partitions the datasets accordingly. UTXOs with low access rates

will be stored on disk, and those with high access rates will be stored in cache, aiming to reduce memory usage and block verification time. Next, we describe the storage service framework in two parts: the predictive model based on address access rate and the UTXO datasets partitioning strategy.

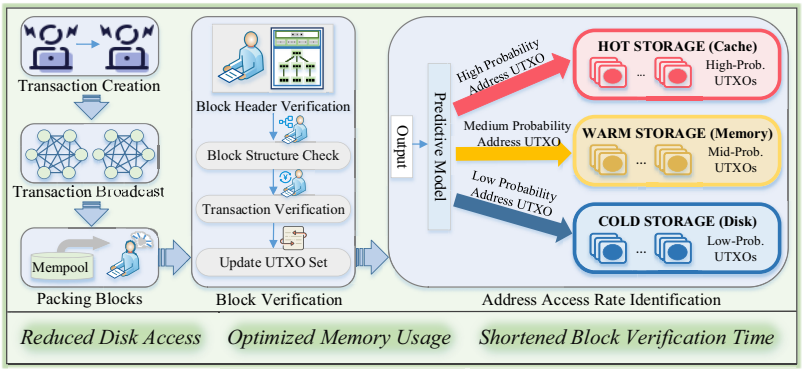

Figure 3: The proposed storage service framework.

4.1.1 Predictive Model

By identifying address access rates, the UTXO datasets can be partitioned more efficiently. Addresses are categorized and stored based on their access probabilities, thereby reducing disk access while increasing cache utilization. Next, how the full node identifies the address access probability for transactions is described. The specific workflow of address access probability identification is shown in Fig. 4. To facilitate the formal description of our probability predictive model, the key variables and parameters used in this section are summarized in Table 1.

After the UTXO dataset has all completed the address access probability identification, for each new block on the blockchain, the full node will carry out address access probability identification for each transaction within the block. The specific process of performing address access probability identification after a new node has accessed the blockchain is described as follows.

(1) Initialization

The full node is initialized and starts the consensus algorithm [35] used by the blockchain system.

(2) Transaction synchronization

After the initialization is complete, the full node is connected to the blockchain system and kept in sync. This includes continuously fetching new blocks from other nodes, verifying and broadcasting them to maintain consistency across the system.

(3) Address access probability identification

The full node optimizes data processing by evaluating the access frequency and transaction interval of UTXO-associated addresses. For a given address a_i, we first extract its average transaction interval $I(a_i)$ and transaction frequency $F(a_i)$ within an observation window T_{win}, normalizing both to $[0,1]$ via min-max normalization. It categorizes addresses as "active" ($A(a_i) = 1$) or "inactive" ($A(a_i) = 0$) based on transaction time intervals, with addresses having recent transactions (time since last transaction $\leq T_{th}$) labeled as active. It then applies a probabilistic classification combining transaction frequency. Specifically, the joint probability is quantified as

$$P_{access}(a_i) = \alpha \cdot A(a_i) \cdot (1 - I(a_i)) + \beta \cdot (1 - A(a_i)) \cdot F(a_i) \tag{1}$$

where α and β are weight coefficients ($\alpha + \beta = 1$) used to balance the influence of recent activity and historical transaction frequency. To determine their optimal values without introducing computationally expensive machine learning models unsuitable for resource-constrained full nodes, we conducted an empirical grid search during preliminary simulations. We systematically varied α and β in increments of 0.1 and evaluated the resulting cache hit rates and block verification latencies. Because IoT workloads typically exhibit strong temporal locality—meaning recent transaction activity is a significantly stronger predictor of immediate future access than overall historical frequency—the weight for the time interval (α) requires a higher proportion. Our empirical results demonstrated that setting $\alpha = 0.7$ and $\beta = 0.3$ yields the optimal performance trade-off, achieving high-precision UTXO access probability prediction with minimal computational overhead. Based on this predicted probability, addresses with $P_{access}(a_i) \geq P_{high}$ are assigned "high probability address"; addresses with $P_{access}(a_i) < P_{low}$ are labeled "low probability address"; and all other addresses are classified as "medium probability address". These dynamic thresholds (P_{high} and P_{low}) are adaptively adjusted according to the remaining memory size of the full node. All intensive computations are offloaded to full nodes, ensuring no overhead on IoT devices.

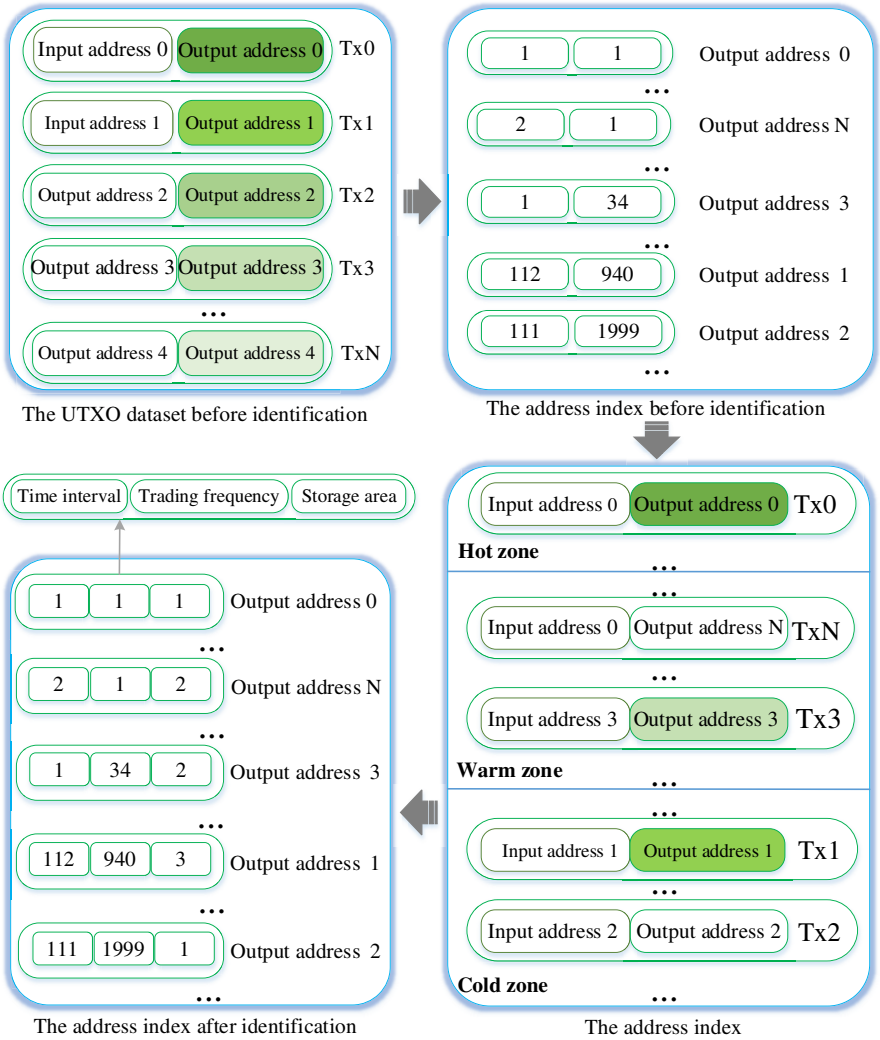

Figure 4: Address access probability identification process.

Table 1: Symbol definition for probability prediction model.

Symbol	Definition	Unit
T_{win}	Observation time window for transaction behavior feature extraction	s
$I(a_i)$	Normalized average transaction interval of address a_i	–
$F(a_i)$	Normalized transaction frequency of address a_i in the observation window	–
T_{th}	Activity threshold for address classification	–
$A(a_i)$	Binary activity label of address a_i (1 for active, 0 for inactive)	–
$P_{access}(a_i)$	Predicted access probability of the UTXO associated with address a_i	–
S_r	Remaining available memory size of the full node	Byte
S_o	Memory size required to store all new UTXOs in a block	Byte

It is important to note that resource-constrained IoT end-devices only act as lightweight clients that generate and broadcast transactions; they do not perform address indexing or probability identification. These computationally intensive tasks are entirely offloaded to the full nodes in the blockchain network, thereby avoiding any CPU strain or performance degradation on the IoT devices themselves.

When the remaining memory size S_r of the full node is greater than or equal to the memory size S_o required for storing UTXOs in the new block, i.e., $S_r \geq S_o$, the full node performs the address access probability identification for each UTXO generated by the new block. The full node obtains the region where the address is located by the address index and identifies the address access probability based on the storage region field. When the storage region field is 1, it is considered a hot address; and when the region is 2 or 3, it should be considered a warm or cold address. The interval time field and transaction frequency field of the address index are also updated. Conversely, when $S_r < S_o$, the full node will initiate a new round of address access probability identification for the UTXO datasets, aiming to improve the match between the UTXO datasets and the address access probability.

4.1.2 UTXO Datasets Partitioning Strategy

After address access probability identification, low-probability addresses are sorted by descending transaction frequency, with the top N UTXOs moved to the cold zone (disk). High-probability addresses are sorted by ascending frequency (prioritizing the most recent transaction time when compare same probabilities), with the first n UTXOs stored in the hot zone (memory cache) and the remainder in the warm zone (memory).

(1) UTXOs deletion

After completing the partition of the UTXO datasets based on address access probability identification, the full node performs verification for each transaction within the added block. Subsequently, it performs access probability identification and partitioning for the output addresses of these transactions. Since the block header verification and script verification in block verification are not affected, we will focus on the specific flow of deleting UTXO, as shown in Fig. 5.

For each transaction, the full node verifies that its input is valid. This includes checking whether the input referenced by the transaction is in the UTXO datasets and has not already been used by another transaction. First, the full node obtains the transaction ID referenced in the transaction input and output index to determine which output of which transaction is referenced by that input. Second, according to the referenced transaction ID and output index, the full node looks up the corresponding transaction output in the UTXO datasets to check whether the transaction output exists, i.e., whether it has not been spent yet.

If the corresponding transaction output is found in the UTXO datasets and has not been spent, then the transaction input is valid; otherwise, the transaction input is invalid and the node will reject the transaction and the block. Finally, if the transaction input is valid, the full node marks the corresponding UTXO as spent to prevent the UTXO from being reused by other transactions. After verifying the inputs of all transactions, the full node updates the UTXO collection and removes the UTXOs that have been used.

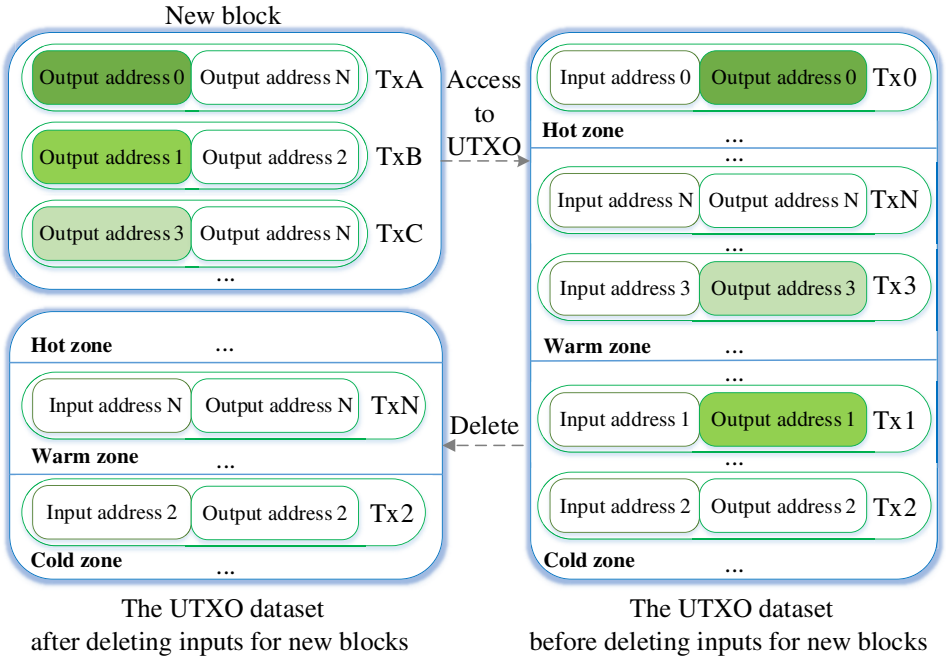

Figure 5: Deletion process of input UTXO in the new block.

The time taken by the full node to perform block verification is directly proportional to the time taken by the full node to perform UTXO access, when the time taken by the UTXO access is less, the time taken by the full node to complete block verification will be shorter.

(2) UTXOs addition

The workflow of adding the new UTXOs is shown in Fig. 6. The full node removes the inputs for each transaction from the UTXO datasets after completing the new block verification. At the same time, the full node will identify the address access probability of the output of each transaction and partition it by looking at the storage area field on the address index. If the address associated with the UTXO is identified as a hot address, the UTXO will be stored directly in the cache; if it is recognized as a cold address, it will be stored directly on disk; and if it is recognized as a warm address, it will be stored in memory.

While dynamically migrating UTXOs between the hot, warm, and cold zones introduces a minor computational overhead, this operation occurs entirely within memory. This negligible overhead is heavily outweighed by the massive latency reductions achieved by avoiding disk I/O operations during script verification. Consequently, the multi-zone migration does not cause synchronization delays; rather, it actively accelerates the overall transaction processing and node synchronization.

Beyond latency improvements, this architectural choice also yields significant energy benefits. The design of the multi-zone framework is motivated by the pronounced energy disparity between memory and disk operations, as recently reported in edge and IoT-oriented energy efficiency studies. In resource-constrained IoT gateways, a disk I/O operation consumes orders of magnitude more power than a CPU

cache query or memory access. By utilizing a lightweight predictive model to retain high-probability UTXOs in the Hot and Warm zones, DASS effectively reduces the activation frequency of high-power disk interfaces. As a result, the framework achieves a strategic trade-off: a modest increase in CPU computation leads to substantial net energy savings at the system level, consistent with recent IoT-blockchain energy optimization findings.

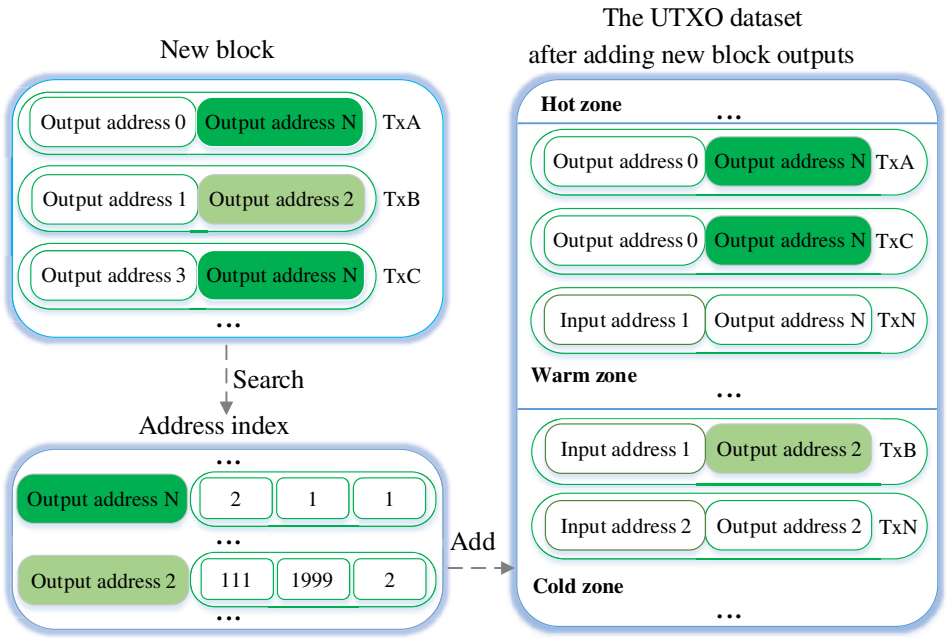

Figure 6: Addition process of generated UTXO.

4.2 Theoretical Analysis of the Multi-Zone Service Framework

In this section, we establish a rigorous theoretical model for the DASS framework, including block verification latency, memory storage optimization, and proof of optimization effectiveness. To ensure clarity and consistency throughout the subsequent mathematical derivations, the primary notations related to time, storage, and energy evaluation are described in Table 2. All derivations are based on the following explicit premise assumptions, which are consistent with real-world UTXO-based blockchain systems:

1. The access latency of a single UTXO follows a fixed hierarchical order: $t_{ACachei} \ll t_{AMi} \ll t_{ADi}$, which is validated by real hardware measurements in existing research.
2. The total block verification time is linearly correlated with the total UTXO access time, as script verification can be accelerated via signature cache and is not affected by UTXO storage location.
3. Blockchain transactions follow the input-output balance principle: the number of consumed UTXOs (inputs) and generated UTXOs (outputs) in a block are statistically stable for a given workload.

4.2.1 Block Verification Latency Model

The core overhead of full-node block verification comes from transaction verification, which is dominated by two parts: script verification and UTXO access. For a new block with N transaction inputs, the total block verification time T_{BV} is defined as:

$$T_{BV} = T_{SV} + T_A + T_{others} \tag{2}$$

where T_{SV} is the total script verification time, T_A is the total UTXO access time, and T_{others} is the negligible time overhead for non-core steps (e.g., Merkle tree verification, PoW validation), which is treated as a constant in our model.

Table 2: Key notations for performance and theoretical analysis.

Symbol	Definition	Unit
T_{BV}	Total block verification time	s
T_{SV}	Total script verification time for all transaction inputs	s
T_A, T'_A	Total UTXO access time after/before optimization	s
ΔT_A	Total reduction in UTXO access time after optimization	s
T_{others}	Negligible time overhead for non-core verification steps	s
t_{SVi}	Script verification time for the i-th UTXO input	s
t_{Ap}	Time overhead for single signature cache access	s
t_{ECDSA}	Time overhead for single ECDSA signature verification	s
P_M, P'_M	Probability of a UTXO being stored in memory after/before optimization	–
P_D, P'_D	Probability of a UTXO being stored in disk after/before optimization	–
P_{Cache}	Probability of a UTXO being stored in hot zone (cache)	–
N_{Cache}	Number of UTXO inputs in hot zone (cache) after optimization	–
N_M, N'_M	Number of UTXO inputs in warm zone (memory) after/before optimization	–
N_D, N'_D	Number of UTXO inputs in cold zone (disk) after/before optimization	–
$t_{ACachei}$	Single access latency for the i-th UTXO in cache (hot zone)	μs
t_{AMi}	Single access latency for the i-th UTXO in memory (warm zone)	μs
t_{ADi}	Single access latency for the i-th UTXO in disk (cold zone)	μs
n	Total number of transactions in a new block	–
λ_i, λ'_i	Number of UTXOs consumed from non-cold zones/memory after/before optimization	–
∂_i, ∂'_i	Number of new UTXOs generated and stored in non-cold zones/memory after/before optimization	–
s	Fixed storage size of a single UTXO	Byte
S_B, S'_B	Net change in memory usage after processing a new block (optimized/baseline)	Byte
ΔS_B	Total reduction in memory usage after optimization	Byte
E_{total}, E'_{total}	Total energy consumption of a full node for processing a block after/before optimization	J/Wh
E_{disk}	Total energy consumed by disk I/O operations per block	J/Wh
E_{mem}	Total energy consumed by memory and cache access per block	J/Wh
E_{cpu}	Total CPU energy consumption per block	J/Wh
e_{disk}	Energy consumption of a single disk I/O operation	J
e_{mem}	Energy consumption of a single memory access	J
e_{cache}	Energy consumption of a single cache access	J
e_{base}	Baseline CPU energy consumption for block verification	J
Δe_{dass}	Additional CPU energy overhead introduced by the DASS prediction model	J
ΔE	Net energy savings achieved by the DASS framework	J/Wh

For script verification time T_{SV}, we have:

$$T_{SV} = \sum_{i=1}^{N} t_{SVi} \tag{3}$$

where t_{SVi} is the script verification time for the i-th UTXO input. For transactions already in the node's mempool, script verification can be skipped via signature cache access; otherwise, full ECDSA signature verification is required. Thus, t_{SVi} follows:

$$t_{SVi} = \begin{cases} t_{Ap}, & \text{if the transaction is in the mempool} \\ t_{ECDSA} + t_{others}, & \text{otherwise} \end{cases} \tag{4}$$

where t_{Ap} is the signature cache access time, and t_{ECDSA} is the ECDSA signature verification time. It is important to note that T_{SV} is independent of UTXO storage location, so the optimization of block verification time is entirely determined by the reduction of T_A.

For the baseline system (before optimization), all UTXOs are either stored in memory or disk. Let P'_M be the probability of a UTXO being stored in memory, then the number of memory-resident UTXOs is $N'_M = N \cdot P'_M$, and the number of disk-resident UTXOs is $N'_D = N \cdot (1 - P'_M)$. The total UTXO access time for the baseline system is:

$$T'_A = \sum_{i=1}^{N'_M} t_{AMi} + \sum_{i=1}^{N'_D} t_{ADi} \tag{5}$$

For the DASS-optimized system, UTXOs are partitioned into three hierarchical zones: cache (hot), memory (warm), and disk (cold). Let P_{Cache}, P_M, P_D be the probability of a UTXO being stored in each zone, with $P_{Cache} + P_M + P_D = 1$. The number of UTXOs in each zone is $N_{Cache} = N \cdot P_{Cache}$, $N_M = N \cdot P_M$, $N_D = N \cdot P_D$. The total UTXO access time after optimization is:

$$T_A = \sum_{i=1}^{N_{Cache}} t_{ACachei} + \sum_{i=1}^{N_M} t_{AMi} + \sum_{i=1}^{N_D} t_{ADi} \tag{6}$$

We then derive the total reduction in UTXO access time achieved by DASS:

$$\Delta T_A = T'_A - T_A \tag{7}$$

Substituting (5) and (6) into (7), and using $N = N'_M + N'_D = N_{Cache} + N_M + N_D$, we get:

$$\Delta T_A = \sum_{i=1}^{N_{Cache}} (t_{AMi} - t_{ACachei}) + \sum_{i=1}^{N'_M - N_M - N_{Cache}} t_{AMi} - \sum_{i=1}^{N_D - N'_D} t_{ADi} \tag{8}$$

4.2.2 Memory Storage Optimization Model

Blockchain transactions follow the input-output balance principle: each transaction consumes existing UTXOs (inputs) and generates new UTXOs (outputs). We establish a memory usage model to quantify the storage optimization effect of DASS.

For the baseline system, let λ'_i be the number of memory-resident UTXOs consumed by the i-th transaction, and ∂'_i be the number of new UTXOs stored in memory by the i-th transaction. For a block with n transactions, the net change in memory usage S'_B is:

$$S'_B = \left(\sum_{i=1}^{n} \partial'_i - \sum_{i=1}^{n} \lambda'_i \right) \cdot s \tag{9}$$

where s is the fixed storage size of a single UTXO.

For the DASS-optimized system, we define non-cold zones as the sum of hot (cache) and warm (memory) zones. Let λ_i be the number of non-cold zone UTXOs consumed by the i-th transaction, and ∂_i be the number of new UTXOs stored in non-cold zones. The net change in memory usage S_B after optimization is:

$$S_B = \left(\sum_{i=1}^{n} \partial_i - \sum_{i=1}^{n} \lambda_i \right) \cdot s \tag{10}$$

We then derive the total memory usage reduction achieved by DASS ($\Delta S_B = S'_B - S_B$):

$$\Delta S_B = \left[\sum_{i=1}^{n} (\partial'_i - \partial_i) - \sum_{i=1}^{n} (\lambda'_i - \lambda_i) \right] \cdot s \tag{11}$$

DASS stores only high-probability UTXOs in non-cold zones, which means: (1) the number of new UTXOs stored in memory is reduced ($\partial_i < \partial'_i$); (2) the number of memory-resident UTXOs consumed per transaction is increased ($\lambda_i > \lambda'_i$), as high-probability UTXOs are more likely to be spent. Both effects contribute to a positive ΔS_B.

4.2.3 Proof of Optimization Effectiveness

We now prove the sufficient conditions for DASS to achieve positive performance gains.

Theorem 1 (Latency Optimization Sufficient Condition): If the DASS framework achieves a cache hit rate higher than the memory hit rate of the baseline system, i.e., $P_{Cache} + P_M > P'_M$, then $\Delta T_A > 0$, meaning DASS reduces total UTXO access time.

Proof: From Eq. (7), since $t_{ADi} \gg t_{AMi} \gg t_{ACachei} > 0$, we have:

$$\Delta T_A > \sum_{i=1}^{N_{Cache}+N_M-N'_M} t_{AMi} \tag{12}$$

If $P_{Cache} + P_M > P'_M$, then $N_{Cache} + N_M - N'_M > 0$, so $\Delta T_A > 0$. □

Theorem 2 (Memory Optimization Sufficient Condition): If the DASS framework correctly identifies high-probability UTXOs, such that the consumption rate of non-cold zone UTXOs is higher than the generation rate, i.e., $\sum_{i=1}^{n} \lambda_i > \sum_{i=1}^{n} \partial_i$, then $\Delta S_B > 0$, meaning DASS reduces net memory usage.

Proof: From Eq. (10), for the baseline system, all new UTXOs are stored in memory, so ∂'_i is equal to the total number of outputs, and λ'_i is the number of memory-resident inputs. For the optimized system, $\partial_i < \partial'_i$. If $\sum_{i=1}^{n} \lambda_i > \sum_{i=1}^{n} \partial_i$, the node releases memory after processing the block. Since baseline memory usage typically increases over time, $\Delta S_B = S'_B - S_B > 0$. □

4.2.4 Quantitative Energy Efficiency Model

Although physical power draw varies significantly across different IoT hardware architectures, we establish a quantitative energy model to theoretically validate the energy-efficiency claims of the DASS framework. The total energy consumption for processing a block, denoted as E_{total}, is determined by the CPU processing energy E_{cpu}, memory access energy E_{mem}, and disk I/O energy E_{disk}:

$$E_{total} = E_{cpu} + E_{mem} + E_{disk} \tag{13}$$

For the baseline system, the energy consumption E'_{total} is modeled as:

$$E'_{total} = N \cdot e_{base} + N'_M \cdot e_{mem} + N'_D \cdot e_{disk} \tag{14}$$

where e_{base} is the baseline CPU energy for script verification, and e_{mem} and e_{disk} are the energy consumed by a single memory and disk access, respectively.

For the DASS-optimized system, the probabilistic predictive model introduces a negligible CPU energy overhead Δe_{dass} for calculating the access probability (which operates in strictly $O(1)$ time complexity). The optimized energy consumption is:

$$E_{total} = N \cdot (e_{base} + \Delta e_{dass}) + N_{Cache} \cdot e_{cache} + N_M \cdot e_{mem} + N_D \cdot e_{disk} \tag{15}$$

The net energy savings $\Delta E = E'_{total} - E_{total}$ can be expressed as:

$$\Delta E = (N'_D - N_D) \cdot e_{disk} + (N'_M - N_M) \cdot e_{mem} - N_{Cache} \cdot e_{cache} - N \cdot \Delta e_{dass} \tag{16}$$

According to established computer architecture measurements, disk I/O operations consume several orders of magnitude more energy than Static Random-Access Memory (SRAM) and Dynamic Random-Access Memory (DRAM) accesses (i.e., $e_{disk} \gg e_{mem} > e_{cache}$). Furthermore, the lightweight design of our probability identification algorithm ensures that the additional CPU computational energy is extremely minimal ($\Delta e_{dass} \to 0$). Since the DASS framework significantly reduces the number of disk-resident UTXO accesses ($N_D \ll N'_D$), the energy saved from avoided disk I/O overwhelmingly compensates for the minor prediction overheads. Therefore, $\Delta E \gg 0$, mathematically proving that the proposed framework delivers substantial quantitative energy efficiency for resource-constrained IoT environments.

5 Performance Evaluation

The simulation environment is implemented in Python 3.11.4 and is based on blockchain system principles. Initially, 100,000 UTXOs form the datasets, with a cache set to store n UTXOs occupying N% of memory. Different data partitioning methods are applied: LRU & MRU, LFU & MFU, and DASS. Each method has four memory usage settings (50%, 60%, 70% and 80%) with 400 to 700 cached UTXO items. For input-output partition, 200 UTXOs were selected as inputs, and 150 as outputs, based on time intervals and transaction frequencies. 30% of the output addresses had low access rates. Finally, t_{AMi} was set to 16 μs, t_{ADi} to 10120 μs [20], and $t_{ACachei}$ to 1 μs.

5.1 Dataset Construction and IoT Scenario Adaptability Analysis

To fit the actual situation of the overall datasets, we download the relevant UTXO age distribution data from the Bitcoin browser, as shown in Fig. 7. The vast majority of the UTXOs are older than 1 week, and even occupy more than 97% of the overall UTXO datasets. To better control the overall datasets size,

we generate 90% of UTXOs older than 72 h and 10% of UTXOs not older than 72 h when conducting the simulation experiments.

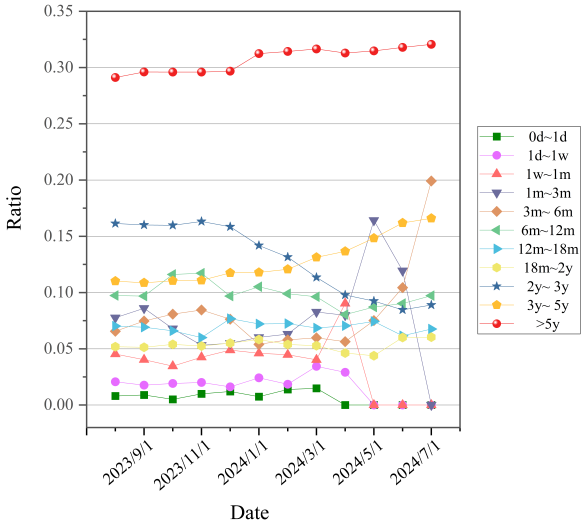

Figure 7: Bitcoin UTXO age distribution.

In this section, the block height of 824334 in Bitcoin is selected for data analysis. We accessed a Bitcoin API to obtain the transaction frequency and the last five transaction times of the related addresses. Because large-scale, real-world UTXO datasets natively generated by IoT environments are currently scarce, we extracted realistic structural features (e.g., transaction frequency, time intervals, and UTXO age distribution) from this block. By parameterizing these features, we synthesized a workload characterized by high-frequency, small-amount transactions to mimic the data generation behaviors of distributed IoT devices. We then applied these real-world probabilistic rules to generate our evaluation dataset of 100,000 UTXOs. This hybrid approach ensures our simulation effectively mimics IoT data generation patterns while maintaining the empirical validity of a genuine UTXO-based blockchain system. By comparing the transaction frequencies of the addresses in the block, Fig. 8 is obtained.

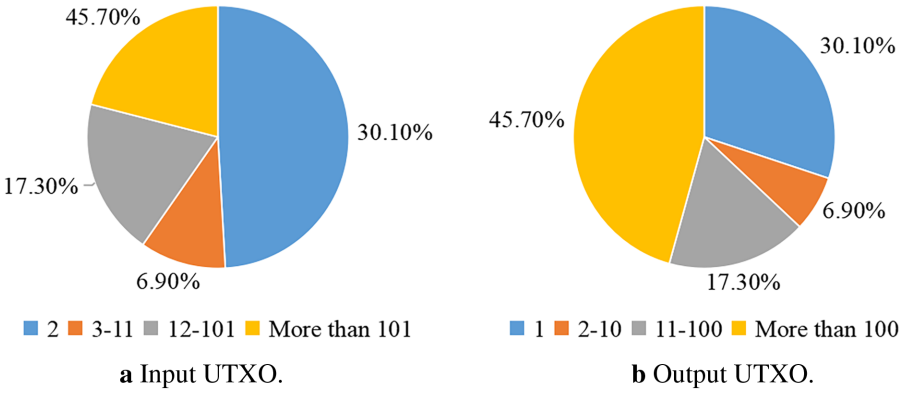

Figure 8: Transaction frequency distribution for UTXO related addresses in block 824334.

The analysis focuses on the transaction frequency of both input and output UTXO addresses. It is found that addresses with a transaction frequency of 2 occupy half of the input UTXO addresses, while addresses

with a transaction frequency of more than 100 occupy half of the output UTXO addresses. Since input represents released UTXOs and output represents new UTXOs, the UTXOs associated with addresses with a transaction frequency of 1 have a higher probability of being packaged by the miner. This is because the address space with the transaction frequency of 1 is small. The purpose of storing the UTXOs associated with addresses with this characteristic in the cache is to increase the cache hit rate, thereby reducing memory usage by accessing and releasing hit UTXOs.

After a detailed comparison of the transaction time difference and transaction frequency under the same address in the input UTXOs, Fig. 9 is obtained. It is found that among the addresses associated with the input UTXOs packed in the block, about 70% of the addresses have completed transactions in the last 72 h. For the remaining 30% of the addresses, the address distribution is related to the transaction frequency of the address. As the frequency of transactions increases, the UTXOs associated with that address are less likely to be spent. UTXOs with this characteristic can be moved from memory to disk to reduce the number of disk access during block verification, thus reducing block verification time.

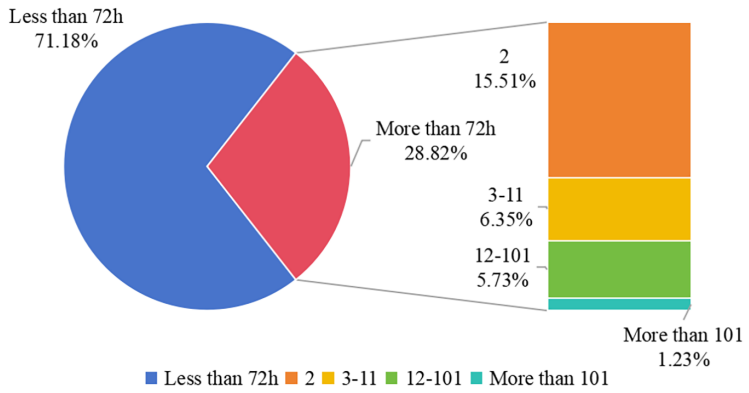

Figure 9: Distribution of time intervals and transaction frequencies for input UTXO related addresses.

5.2 Performance Comparisons on UTXO Access Time

In this section, the UTXO access time evaluation experiments and the number of UTXO access in hot/cold zone experiments are performed on the newly generated block. The results of the experiments record the UTXO access time, the number of cache and disk UTXO accesses spent on verifying the new blocks under different partitioning methods.

As disk access is relatively slow, an increase in UTXO accesses leads to longer block verification time. The UTXO access time can be reduced by minimizing the number of disk access. Fig. 10 compares disk UTXO accesses across different partitioning methods after a new block is generated. Regardless of cache capacity and memory usage, the DASS results in the fewest disk accesses. Specifically, with memory usage ranging from 50% to 80%, the DASS requires 1~14 disk accesses, while the LFU and LRU methods require 103~141, and 13~35 accesses, respectively.

Generally speaking, the cache is the fastest to access. When the number of cache UTXO accesses is larger, it means that the number of UTXOs stored in other zones is smaller. At this point, the access time needed for block verification is also reduced. Fig. 11 shows the comparison of cache UTXO accesses under different partitioning methods after a new block is generated. It can be seen that the cache UTXO accesses in the DASS are the largest regardless of the cache capacity and memory usage.

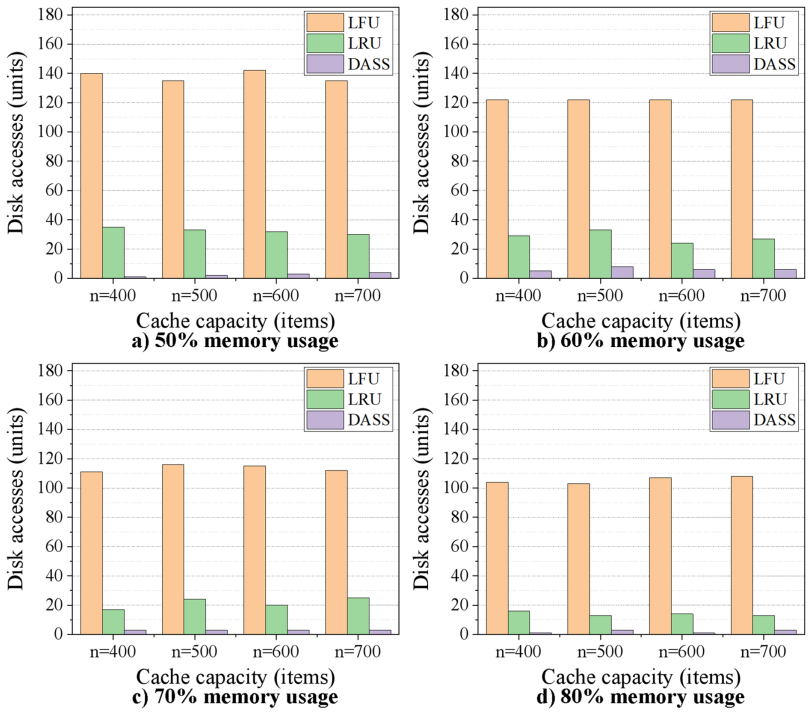

Figure 10: Disk UTXO access comparisons.

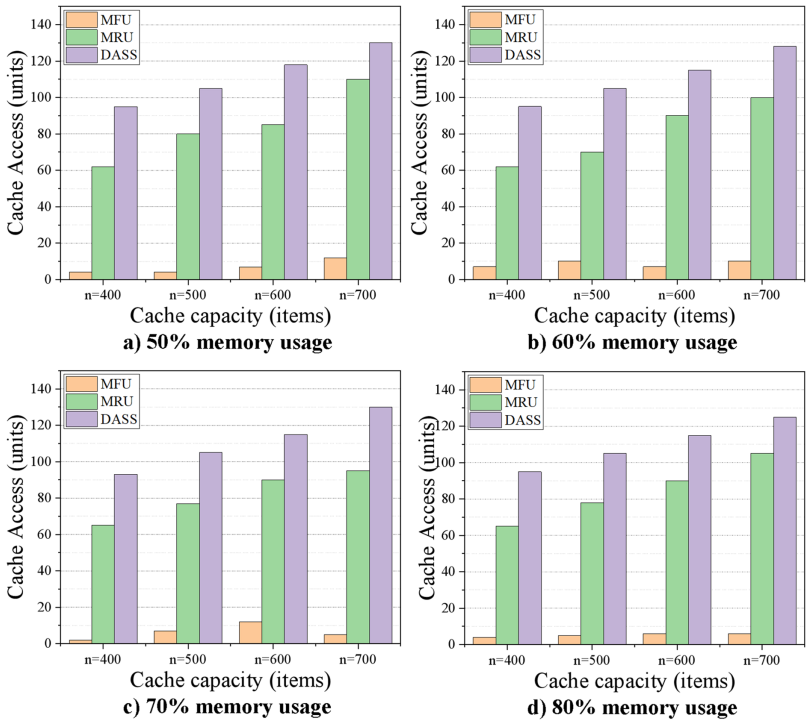

Figure 11: Number comparisons of cache UTXO access.

Specifically, the DASS caches about 94~130 UTXO accesses when the memory utilization is 50%~80%, while the number of cache UTXO accesses under the MFU and MRU methods are 2~11 and 63~111, respectively.

Fig. 12 compares the UTXO access time for different partitioning methods, showing that the proposed method outperforms the LRU & MRU and LFU & MFU methods in reducing access time, regardless of cache size and memory usage. The proposed DASS requires 0.01~0.14 s for UTXO access when memory usage is 50%~80%, while the LFU & MFU and LRU & MRU methods require 1.04~1.44 s and 0.13~0.39 s, respectively. The accuracies of other partitioning methods are lower than the DASS due to the lack of correlation when using transaction frequency and recent transaction time alone.

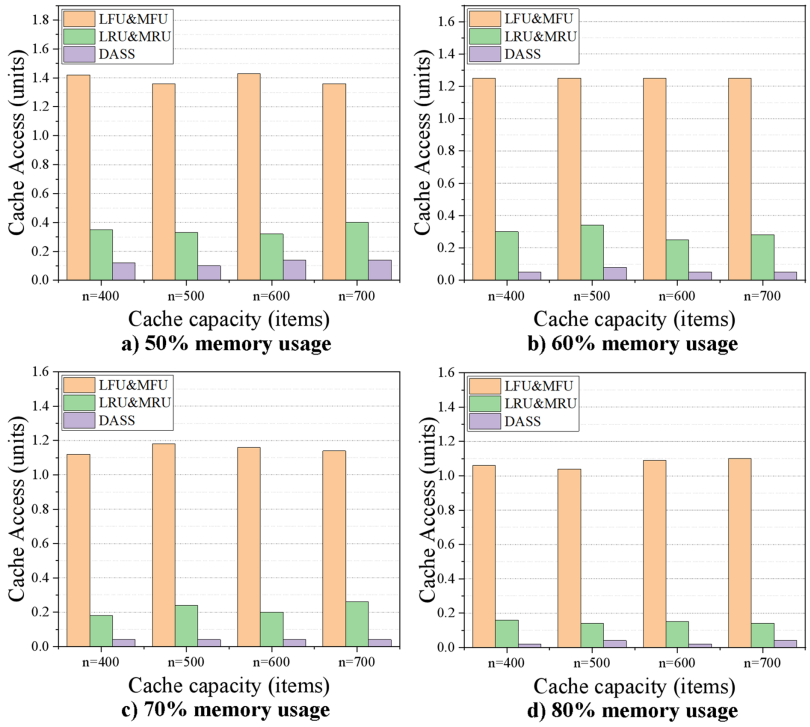

Figure 12: UTXO access time comparisons.

Based on the above experimental evaluation and comparisons, the method proposed in this paper can achieve efficient data management and reduce UTXO access time.

5.3 Performance Comparisons on Memory Storage

In this part, performance experiments are conducted to evaluate the total memory release for new blocks. The experimental results record the total number of UTXOs released by different partitioning methods after releasing the input of the new block and storing the output of the new block. In this case, the output of the new block is set to contain 30% of addresses with a low address access probability, resulting in Fig. 13. When the number of memory accesses and releases is higher, it means that there is more space left in the memory, thereby reducing more memory usage.

As shown in Fig. 13, the method proposed in this paper provides better performance in freeing up memory size compared to the LRU & MRU and LFU & MFU partitioning methods for different cache sizes and memory usage settings. When the memory usage is 50%~80%, the proposed DASS can free up about

72~86 UTXOs, and the LRU & MRU can free up about 39~55 UTXOs. However, with the LFU & MFU, not only is there no space left in the memory, but additional space is needed to store the excess 6~29 UTXOs. This is because the accesses and releases of the cached UTXOs were optimized to the maximum before, and therefore, the overall memory release is optimized as well, thus reducing the memory usage.

Based on the above experimental evaluation and analysis, the method proposed in this paper can achieve efficient data storage and reduce the memory size required for block storage.

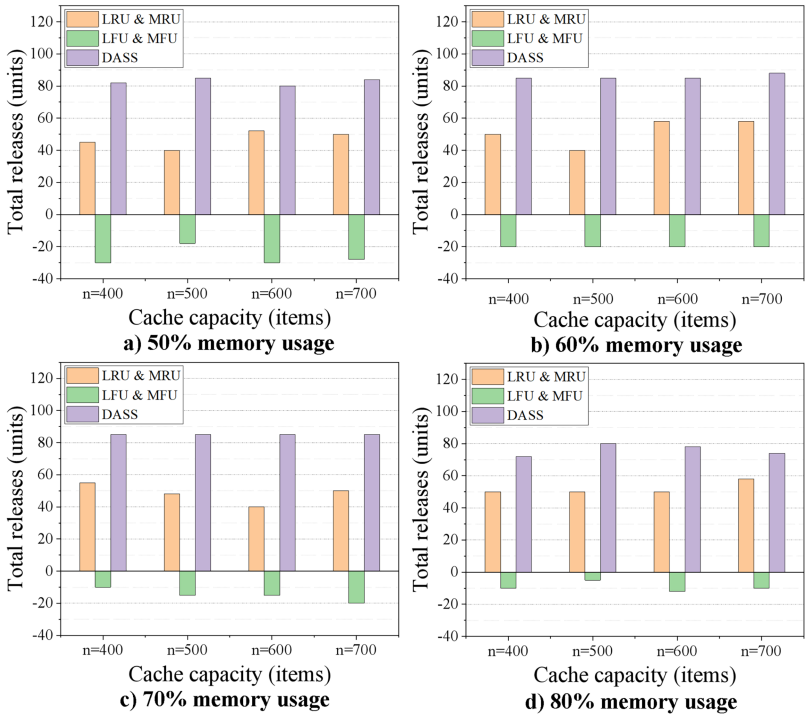

Figure 13: Total number comparisons of memory UTXO releases.

5.4 Performance Comparisons on Block Verification Time

For the designed simulations, block verification time evaluation experiments are performed on newly generated blocks. The experimental results record the total time spent verifying the same new block under different verification optimization methods. Among them, the PUCO [36] method utilizes the cache to store the script data needed by the exchange for signature verification and reduces the block verification time by accelerating the script verification, while the rest of methods [37] utilize the cache to store the relevant UTXO data and reduce the block verification time by accelerating the UTXO access.

Fig. 14 shows that the proposed method outperforms LRU & MRU, LFU & MFU, and PUCO methods in reducing block verification time under various memory usage and datasets conditions. When memory usage is 60%~90%, the proposed DASS requires 0.2~1.9 s for block verification, while PUCO, LRU & MRU, and LFU & MFU methods take 0.8~4.1 s, 0.6~2.5 s, and 1.1~6.1 s, respectively. The proposed method accelerates block verification by 2.2~4 times compared to PUCO, 1.3~3 times compared to LRU & MRU, and 3.2~5.5 times compared to LFU & MFU. This demonstrates the method's high efficiency in data processing and block verification time reduction.

While this study successfully quantifies steady-state block verification time, the DASS framework also presents theoretical benefits for Initial Block Download (IBD) synchronization. Since IBD requires

a new node to fetch and validate the entire historical chain, maintaining an optimized cache of high-probability active UTXOs can significantly reduce the disk access bottlenecks traditionally experienced by new nodes joining the network. Quantifying the exact IBD latency reductions remains a subject for our future empirical studies.

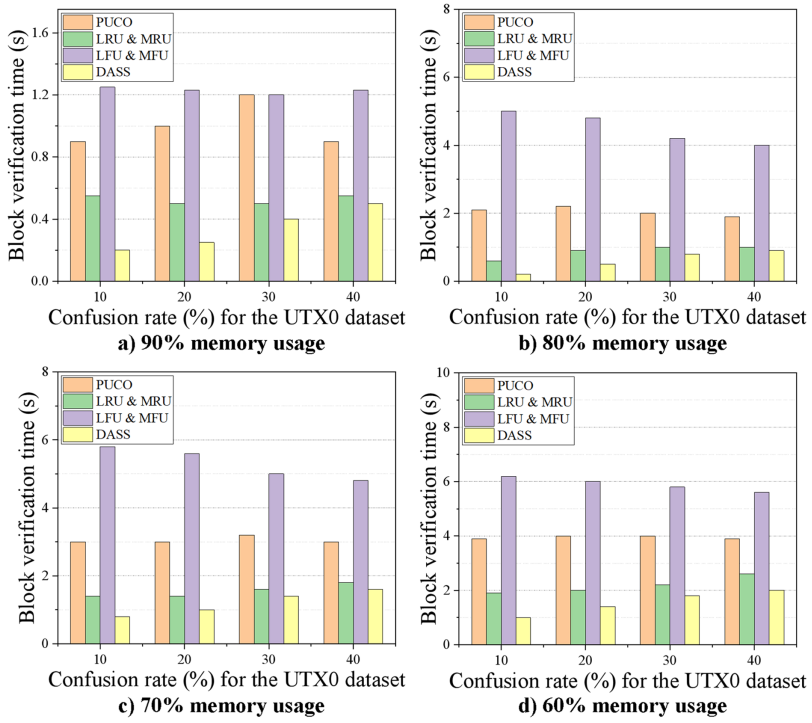

Figure 14: Block verification time evaluation.

5.5 Transferability to Real IoT Transaction Scenarios

While our evaluation is conducted on a synthetic workload parameterized from Bitcoin statistics, it is crucial to analyze how these performance gains transfer to real-world IoT blockchain scenarios (e.g., smart grid monitoring or industrial IoT).

In the Bitcoin network, transaction behaviors are predominantly human-driven, resulting in high randomness in transaction intervals and frequencies. In contrast, typical IoT workloads are machine-driven. IoT end-devices usually follow pre-programmed logic, exhibiting strict periodicity (e.g., sensors reporting data every 10 min) and fixed communication pairs (e.g., terminal devices exclusively sending data to specific edge gateways). The core of our proposed DASS framework is the predictive model formulated in Section 4.1.1, which fundamentally relies on capturing the data association between transaction time intervals $I(a_i)$ and frequencies $F(a_i)$. For highly periodic IoT workloads, the variance of transaction intervals approaches zero, making the address activity highly predictable. Our prediction model intrinsically rewards this regularity, enabling it to accurately lock high-probability UTXOs into the cache zone before periodic sensor data arrives.

Consequently, the chaotic nature of the Bitcoin-parameterized dataset actually presents a more challenging prediction environment for our model. The significant performance improvements observed in Sections 5.2 to 5.4 (e.g., cache hit rates and latency reductions) can be considered a conservative estimate, or a performance lower bound. When transferred to highly regular real-world IoT transaction patterns, the

prediction accuracy of the DASS framework is theoretically guaranteed to increase, thereby yielding even more substantial reductions in disk access and block verification latency.

6 Conclusion

In this paper, we propose an improved blockchain storage service that utilizes data association to construct a multi-zone storage framework for UTXO-based blockchain systems, significantly improving block verification latency and memory utilization in resource-constrained IoT environments. First, we analyze the transaction time intervals and address-specific frequencies of UTXO transactions using real Bitcoin data to capture the probabilistic features inherent in IoT device transactions. Then, an address identification model based on address access probability is proposed based on the probability features. Next, the UTXO datasets are identified using the address access probability identification model, and higher or lower address access rates are partitioned and stored for better storage and verification optimization. In this work, theoretical analysis and a large number of experiments are carried out, and the experimental evaluation shows our proposed UTXO-based transaction processing service effectively reduces block verification time for full nodes while optimizing memory utilization. Furthermore, our adaptability and transferability analysis demonstrates that the highly periodic nature of real IoT workloads will further amplify the prediction accuracy and storage optimization benefits of our framework. To further strengthen the practical deployment of this work, our immediate future research will focus on engineering this data association-based storage service as a pluggable middleware. By integrating it into the storage engine of real UTXO-based blockchain clients, we aim to evaluate its end-to-end performance under real-world IoT physical network conditions. In the future, we also plan to apply this predictive storage service to account-based blockchain systems. By shifting our probability model from UTXO age to account state access frequencies, we aim to alleviate the state bloat scalability challenges inherent in account-based models, further adapting the framework for large-scale, high-throughput IoT deployment scenarios.

Acknowledgement: This work is supported by State Grid Hunan Electric Power Co., Ltd. research project of Full-Link Data Lineage Tracking and Hunan Key Laboratory for Internet of Things in Electricity.

Funding Statement: This research was funded by State Grid Hunan Electric Power Co., Ltd. research project of Full-Link Data Lineage Tracking (No. 5216A624000L).

Author Contributions: Study plan formulation, energy statistics collection and research paper writing, Bin Fang, Qi Yu and Jingyu Zhang; experimental platform construction, performance evaluation and research paper editing, Han Wu and Xingxing Hou; testing suggestions provision, guidance and manuscript proofreading, Han Wu and Jingyu Zhang. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The Bitcoin blockchain data (including the UTXO dataset) analyzed during the current study are publicly available via the official Bitcoin Core repository at https://bitcoincore.org/en/download/.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Yánez W, Bahsoon R, Zhang Y, Kazman R. Architecting internet of things systems with blockchain: a catalog of tactics. ACM Trans Softw Eng Methodol. 2021;30(3):35 doi:10.1145/3442412.
2. Yang Y, Guan Z, Wan Z, Weng J, Pang H, Deng RH. PriScore: blockchain-based self-tallying election system supporting score voting. IEEE Trans Inf Forensics Secur. 2021;16:4705–20. doi:10.1109/TIFS.2021.3108494.

3. Nguyen CT, Nguyen DN, Hoang DT, Pham H, Tuong NH, Xiao Y, et al. BlockRoam: blockchain-based roaming management system for future mobile networks. IEEE Trans Mob Comput. 2022;21:3880–94. doi:10.1109/TMC.2021.3065672.
4. Androulaki E, Barger A, Bortnikov V, Cachin C, Christidis K, Caro AD, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. arXiv:1801.10228. 2018.
5. Nakamoto S. Bitcoin: a peer-to-peer electronic cash system. Bitcoin. 2008;4(2):15. [cited 01 May 2026]. Available from: https://bitcoin.org/bitcoin.pdf.
6. Lombrozo E, Lau J, Wuille P. Segregated witness (consensus layer). Bitcoin Core Dev Team Tech Rep BIP 141. 2015 [cited 01 May 2026]. Available from: https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki.
7. Rizun PR. A transaction fee market exists without a block size limit. Block Size Limit Debate Working Paper. 2015 [cited 01 May 2026]. Available from: https://montreal2015.scalingbitcoin.org/papers/transaction-fee-market-exists-without-block-size-limit.pdf.
8. Poon J, Dryja T. The bitcoin lightning network: scalable off-chain instant payments; 2016 [cited 01 May 2026]. Available from: https://lightning.network/lightning-network-paper.pdf.
9. Singh A, Click K, Parizi RM, Zhang Q, Dehghantanha A, Choo KKR. Sidechain technologies in blockchain networks: an examination and state-of-the-art review. J Netw Comput Appl. 2020;149(7):102471. doi:10.1016/j.jnca.2019.102471.
10. Song T, Shudo K. Block pruning with UTXO aggregation. In: Proceedings of the 2022 IEEE International Conference on Blockchain (Blockchain); 2022 Aug 22–25; Espoo, Finland. p. 312–9.
11. Nguyen VH, Trang HS, Nguyen QT, Huynh-Tuong N, Le TV. Building mathematical models applied to UTXO selection for objective transactions. In: Proceedings of the 2018 5th NAFOSTED Conference on Information and Computer Science (NICS); 2018 Nov 23–24; Ho Chi Minh City, Vietnam. p. 160–4.
12. Zhang J, Zhong S, Wang J, Yu X, Alfarraj O. A storage optimization scheme for blockchain transaction databases. Comput Syst Sci Eng. 2021;36(3):521–35. doi:10.32604/csse.2021.014530.
13. Ding D, Jiang X, Wang J, Wang H, Zhang X, Sun Y. Txilm: lossy block compression with salted short hashing. arXiv:1906.06500. 2019.
14. Özyilmaz KR, Patel H, Malik A. Split-scale: scaling Bitcoin by partitioning the UTXO space. In: Proceedings of the 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS); 2018 Nov 23–25; Beijing, China. p. 41–5.
15. Eyal I, Gencer AE, Sirer EG, Van Renesse R. Bitcoin-NG: a scalable blockchain protocol. In: Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16); 2016 Mar 16–18; Santa Clara, CA, USA. p. 45–59.
16. Xu Y. Section-blockchain: a storage reduced blockchain protocol. In: Proceedings of the 2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS); 2018 Dec 12–14; Melbourne, VIC, Australia. p. 115–25.
17. Feng X, Ma J, Wang H, Wen S, Xiang Y, Miao Y. Space-efficient storage structure of blockchain transactions supporting secure verification. IEEE Trans Cloud Comput. 2022;11(3):2631–45. doi:10.1109/TCC.2022.3220664.
18. Bünz B, Fisch B, Szepieniec A. Transparent SNARKs from DARK compilers. In: Advances in Cryptology—EUROCRYPT 2020. Berlin/Heidelberg, Germany: Springer; 2020. p. 677–706. doi:10.1007/978-3-030-45721-1_24.
19. Matzutt R, Kalde B, Pennekamp J, Drichel A, Henze M, Wehrle K. CoinPrune: shrinking Bitcoin's blockchain retrospectively. IEEE Trans Netw Serv Manag. 2021;18(3):3064–78. doi:10.1109/TNSM.2021.3073270.
20. Jiang S, Li J, Gong S, Yan J, Yan G, Sun Y, et al. BZIP: a compact data memory system for UTXO-based blockchains. J Syst Archit. 2020;109(7):101809. doi:10.1016/j.sysarc.2020.101809.
21. Xu C, Zhang C, Xu J. vChain: enabling verifiable Boolean range queries over blockchain databases. In: SIGMOD '19: Proceedings of the 2019 International Conference on Management of Data. New York, NY, USA: The Association for Computing Machinery (ACM); 2019. p. 141–58. doi:10.1145/3299869.3300083.

22. Dai X, Xiao B, Xiao J, Jin H. An efficient block validation mechanism for UTXO-based blockchains. In: Proceedings of the 2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS); 2022 May 30–Jun 03; Lyon, France. p. 1250–60. doi:10.1109/IPDPS53621.2022.00124.
23. Wang J, Ou W, Alfarraj O, Tolba A, Kim GJ, Ren Y. Block verification mechanism based on zero-knowledge proof in blockchain. Comput Syst Sci Eng. 2023;45(2):1805–19. doi:10.32604/csse.2023.029622.
24. Dryja T. Utreexo: a dynamic hash-based accumulator optimized for the Bitcoin UTXO set. IACR Cryptol ePrint Arch. 2019;2019:611. [cited 01 May 2026]. Available from: https://eprint.iacr.org/2019/611.
25. Bünz B, Kiffer L, Luu L, Zamani M. FlyClient: super-light clients for cryptocurrencies. In: Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP); 2020 May 18–21; San Francisco, CA, USA. p. 928–46. doi:10.1109/SP40000.2020.00049.
26. Wu X, Jiang J, Li X, Cheng J, Meng T. Hierarchical classified storage and incentive consensus scheme for building IoT under blockchain. J King Saud Univ Comput Inf Sci. 2024;36(5):102075. doi:10.1016/j.jksuci.2024.102075.
27. Peng K, Hu X, Hu J, Yao Z, Deng T, Hu M, et al. Resource-efficient joint clustering and storage optimization for blockchain-based IoT systems. Future Gener Comput Syst. 2026;179(6):108354. doi:10.1016/j.future.2025.108354.
28. Wang J, Huang G, Sherratt RS, Huang D, Ni J. Data secure storage mechanism for IIoT based on blockchain. Comput Mater Contin. 2024;78(3):4029–48. doi:10.32604/cmc.2024.047468.
29. Yu J, Zhang X, Wang J, Zhang Y, Shi Y, Su L, et al. Robust and trustworthy data sharing framework leveraging on-chain and off-chain collaboration. Comput Mater Contin. 2024;78(2):2159–79. doi:10.32604/cmc.2024.047340.
30. Oyenuga SA, Ubochi BC, Onuoha O, Nwulu N. A blockchain-based framework for improving energy efficiency and scalability in IoT networks. Int J Educ Manag Eng. 2025;15(5):53–62. doi:10.5815/ijeme.2025.05.05.
31. Habibullah SM, Alam S, Ghosh S, Dey A, De A. Blockchain-based energy consumption approaches in IoT. Sci Rep. 2024;14(1):28088. doi:10.1038/s41598-024-77792-x.
32. Haque EU, Shah A, Iqbal J, Ullah SS, Alroobaea R, Hussain S. A scalable blockchain based framework for efficient IoT data management using lightweight consensus. Sci Rep. 2024;14(1):7841. doi:10.1038/s41598-024-58578-7.
33. Zhang J, Yang J, Li L, Nian Q, Luo L, Guo D. DBUP: dynamic blockchain UTXO processing for storage efficiency optimization. Comput Netw. 2024;254(1):110744. doi:10.1016/j.comnet.2024.110744.
34. Wang X, Chen Y, Zhang Q. Incentivizing cooperative relay in UTXO-based blockchain network. Comput Netw. 2021;185(SI):107631. doi:10.1016/j.comnet.2020.107631.
35. Zhang J, Sun Y, Guo D, Luo L, Li L, Nian Q, et al. A reputation awareness randomization consensus mechanism in blockchain systems. IEEE Internet Things J. 2024;11(20):32745–58. doi:10.1109/JIOT.2024.3408846.
36. Niu Y, Li H, Zhang C, Wei L. Prediction-based UTXO cache optimization for Bitcoin lightweight full nodes. In: Proceedings of the 2021 IEEE Global Communications Conference (GLOBECOM); 2021 Dec 7–11, Madrid, Spain. p. 1–6.
37. Li C, Wu M, Liu Y, Zhou K, Zhang J, Sun Y. SS-LRU: a smart segmented LRU caching. In: DAC '22: Proceedings of the 59th ACM/IEEE Design Automation Conference. New York, NY, USA: The Association for Computing Machinery (ACM); 2022. p. 397–402. doi:10.1145/3489517.3530469.