Auction-Based Storage Resource Allocation for Blockchain

Rui Pan, Yikun Hu, Chubo Liu, *Member, IEEE*, Keqin Li, *Fellow, IEEE*, and Kenli Li, *Senior Member, IEEE*

***Abstract*—The blockchain establishes trust by maintaining a distributed appending-only ledger, which is widely applied to the nodes lacking trust in the edge environment. However, the full-replication storage mode of blockchain is a big challenge for resource-constrained edge devices. What is worse, system performance is also affected by the large storage overhead. Existing solutions to reduce blockchain storage overhead often require additional security assumptions, lack incentives, or fail to account for resource heterogeneity. To overcome these limitations, we design an auction-based storage resource allocation scheme. Winners are selected to store blocks, taking into account the block preferences of nodes, and the fairness of the system. Nodes are incentivized by implementing fairness and equity in distributed auctions and data transactions through smart contracts. Finally, extensive experiments show 65%–81% savings in storage overhead compared to fully replicated storage.**

***Index Terms*—Auction, blockchain, block offloading, distributed system, fair trading.**

I. Introduction

A. Motivation

IN THE age of the Internet of Everything, edge devices generate or collect data and provide diverse services for users [1], [2]. Due to the distrust between edge devices, the potential value of business data cannot be effectively exploited through sharing [3]. While the introduction of trusted third parties ensures collaboration, it will face the risk of single points of failure and privacy leaks.

Blockchain presents an emerging computing paradigm that allows mutually untrusted members to collectively manage a consistent ledger of transactions in a decentralized manner [4]. By deploying blockchain in the edge environment, it provides a decentralized collaboration platform to support multiple edge services [5]. Blockchain's data invariance, enhanced security, and traceability avoid repudiation by malicious nodes and facilitate trust building.

Nevertheless, the huge resource overhead and low performance of blockchain is an urgent challenge for edge environments. There has been a lot of work on solving the throughput bottlenecks of blockchain in terms of consensus algorithms [6], broadcast protocols [7], and transaction execution methods [8]. The elephant in the room is that the throughput boost of blockchain will result in a massive growth of storage overhead. The blockchain replicates data over all nodes thus implementing state machine replication. This fully replicated storage leads to the emergence of capacity bottlenecks. For example, the data volume of Ethernet is currently over 800 GB. The huge storage overhead will result in limiting the participation of edge devices, which weakens decentralization and threatens the security of blockchain. Therefore, how to reduce the storage cost of blockchain is necessary for further adoption of blockchain, especially for resource-constrained edge environments.

A natural optimization approach is to reduce the storage redundancy of the nodes while ensuring the data integrity of the blockchain. We find that different edge nodes have preferences for different block data. For example, edge devices that perform environmental prediction are only interested in blocks that contain temperature and humidity data. It is inappropriate and unrealistic for these nodes to waste a large amount of storage space to store data that is not relevant to their interests. We suggest storing only a portion of block data on resource-constrained edge nodes. However, the design of the storage allocation scheme faces several key issues.

1) The reduction of storage redundancy may lead to the loss of block data due to malicious nodes.
2) Nodes with incomplete copies of data require additional data communication with other nodes. There is a tradeoff between storage redundancy and network overhead.
3) Edge nodes have different storage resources, which should be utilized fairly and fully.
4) The allocation scheme should meet the interests of each node for motivating the nodes to participate.

B. Related Work

Current state-of-the-art works do not fully consider the above key issues. Nakamoto [9] proposed to divide nodes into full nodes and light nodes in Bitcoin white paper. Light nodes store only block headers, which saves a lot of storage

Manuscript received 22 October 2022; revised 18 June 2023 and 20 July 2023; accepted 10 August 2023. Date of publication 14 August 2023; date of current version 7 December 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB2700100 and Grant 2021YFB0300300, and in part by the Hunan Science and Technology Innovation Program under Grant 2023GK2003. *(Corresponding authors: Yikun Hu; Chubo Liu.)*

Rui Pan, Yikun Hu, Chubo Liu, and Kenli Li are with the College of Information Science and Engineering and the National Supercomputing Center in Changsha, Hunan University, Changsha 410082, Hunan, China (e-mail: prui@hnu.edu.cn; yikunhu@hnu.edu.cn; liuchubo@hnu.edu.cn; lkl@hnu.edu.cn).

Keqin Li is with the College of Information Science and Engineering and the National Supercomputing Center in Changsha, Hunan University, Changsha 410082, Hunan, China, and also with the Department of Computer Science, State University of New York at New Paltz, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/JIOT.2023.3304810

space. Nevertheless, data integrity is all dependent on the full node, leading to the risk of data loss. More seriously, the number of full nodes is gradually decreasing as the storage overhead grows (currently there are only about 6000 full nodes in Ethereum), exacerbating the degree of decentralization. Another popular solution is for multiple nodes to form a consensus unit (CU) [10], [11]. All nodes in a CU jointly maintain just one complete copy of the block data. In theory, the larger the size of a CU, the lower the storage cost per node. Nevertheless, this also implies more trust assumptions between nodes, which goes against the original purpose of using blockchain: to build trust in an environment of mistrust.

Unlike the work at the protocol layer described above, the alternative is to focus on the storage engine layer without strong trust assumptions. BFT-Store is proposed in [12], which used erasure coding to divide storage partitions in permissioned blockchain. While the storage overhead can be reduced from $O(n)$ to $O(1)$, recovering the data introduces additional communication overhead and computation overhead. Xu et al. [5] proposed a futile transaction filtering algorithm and the invalid blocks that contain only invalid transactions would be offloaded to cloud storage. However, this introduces a new risk to the validation process. Wang et al. [13] introduced ForkBase to efficiently retrieve and eliminate storage redundancy. But the redundant part of the actual block is very limited. Instead of analyzing the data itself, Dai et al. [14] took into account users' preferences for block data. Users store only those transactions that are of interest to them and partial Merkle branches. But no one wants to keep other users' data, leading to too little redundancy and new security issues.

To make fuller use of storage resources, there are several works that study storage allocation for the entire blockchain system. A fair and efficient distribution scheme for edge environments is presented in [15]. How to find the best location for storing transactions and blocks is modeled as the no capacity facility location problem (NP-hard), and an efficient approximation algorithm is given. Zhang et al. [16] further combined the reputation mechanism and storage allocation to achieve a novel reputation-proof blockchain. However, the above methods require the full knowledge of the entire peer-to-peer (P2P) network, which is extremely difficult to obtain in an untrusted and dynamically changing environment.

C. Our Contribution

Our design goal is to allocate storage resources reasonably and fairly without requiring a strong trust assumption. This reduces the storage overhead of the blockchain while ensuring a certain level of security.

In this article, we design an auction-based block storage allocation model. First, in a decentralized blockchain, we implement distributed auctions without a centralized auctioneer through smart contracts. Then, nodes bid on the blocks based on their preferences for them. Storing blocks selectively on more relevant nodes reduces the possibility of subsequent data communication to some extent. Finally, considering the system load balancing and node bids, the auction algorithm determines a batch of winning nodes with minimized storage and communication costs. The winning nodes store the complete block data, while the losing nodes store only the block headers.

In order to attract more nodes to actively participate, the allocation scheme needs to safeguard the interests of each node. The winning nodes contribute storage resources to maintain a portion of the block data, and they will be paid for this. The losing nodes save storage overhead at the cost of them having to pay for subsequent data requests. We achieve fair trading in a distrustful environment. Both parties to the trade are able to be simultaneously subjected to the correct block data and payoff, and neither party has any way to benefit by repudiation.

In summary, our major contributions are as follows.

1) We implement two-stage bidding auctions through smart contracts without trusted third parties and offload the auction computation to the off-chain.
2) We design a second price auction with multidimensional information. On the one hand, the blocks are allocated to the preferred nodes to reduce the frequency of nodes requesting data; on the other hand, blocks are allocated fairly according to the storage space of nodes to achieve load balancing.
3) In order to achieve a fair trading between two parties who do not trust each other, we verify the correctness of the block data through asymmetric encryption and use the smart contract of blockchain to realize cash on delivery.

D. Paper Outline

The remainder of this article is structured as follows. In Section II, the overall system framework is introduced. Section III describes the distributed auction in detail. Section IV describes the fair trading between untrusted nodes. Next, the performance evaluation is performed in Section V. Finally, we conclude our work in Section VI.

II. System Architecture

We deploy the blockchain in an edge environment, where each node corresponds to an edge device. The data generated by the edge devices are preprocessed and uploaded to the blockchain to support a variety of edge services. These basic data are constructed into transactions that modify the state of the blockchain ledger according to different business logic. Each block contains information of multiple transactions, which are stored in each node after consensus.

The resources of edge devices are constrained. To relieve the storage pressure, we design an auction mechanism to allocate blocks to some nodes for storage. Nodes are rewarded for honestly responding to other nodes' block requests, creating a positive incentive.

To describe the data flow more clearly, we divide the core framework of the node into three layers, namely, the API layer, the blockchain layer, and the storage layer (including the block buffer), as shown in Fig. 1.

The API layer provides interfaces for edge applications to interact with the blockchain layer and the storage layer, such as

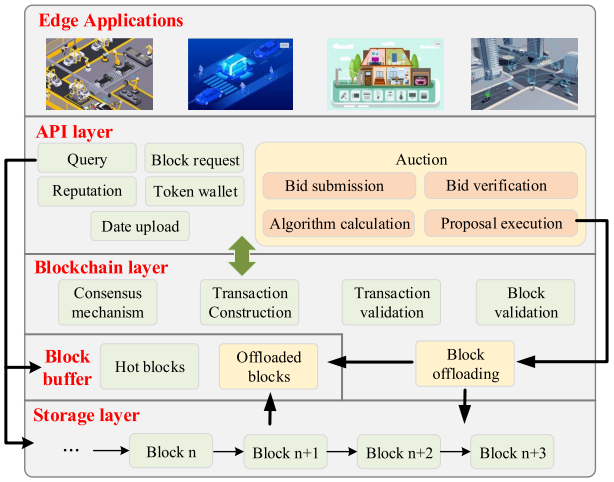

Fig. 1. System architecture.

transaction queries, block requests, data uploads, etc. In addition, a reputation management and auction module is included. Each node will be assigned a reputation value. Reputation is updated in a decentralized manner to prevent manipulation by malicious users. In the auction, nodes make bids on the chain and perform calculations off-chain. Block offloading is performed based on the proposal.

The blockchain layer implements the basic functions of the blockchain. Some operations that change the state of the blockchain ledger, such as data uploading, node bidding, and reputation updates, are constructed as transactions or blocks and broadcasted across the network. The validation module validates the transactions or blocks. The accepted transactions or blocks are stored and the status is updated. The Block offload module analyzes the auction results and sends offload requests to the Storage layer.

The storage layer provides storage services for the upper layers. The block buffer temporarily stores some blocks that are frequently used or about to be unloaded to improve access speed and avoid frequent requests for block data from other nodes.

III. Problem Modeling

There are a total of N blockchain nodes (each node corresponds to an edge device). These heterogeneous nodes have limited storage space and computational resources. We propose an auction-based storage allocation scheme with the following considerations.

1) Each node has different preferences for different blocks of data, e.g., edge devices performing environmental monitoring access temperature and humidity data frequently. It is reasonable to let nodes store data relevant to their own interests. But too little data redundancy carries the risk of data loss. It is necessary to design a global storage allocation scheme.
2) The data request between nodes brings additional communication when the offloaded blocks are needed again. There is a tradeoff between data redundancy and communication overhead when designing the allocation scheme. But it is particularly difficult to obtain full knowledge of P2P networks in an untrusted environment.
3) The node sacrifices storage space and contributes to the overall system because it helps other nodes store block data. We believe that this should not be free, especially in resource-constrained edge environments. Nodes should be rewarded for responding to other nodes' requests for block data.

A. Auction Process

In a decentralized blockchain, we implement distributed auctions between untrusted nodes via smart contracts with the following process.

Step 1 (Select Blocks): The auction starts when the size of the blocks that have been consensused exceeds a threshold. These blocks will be auctioned together.

Step 2 (Submit Bids): In the two-stage bidding phase, nodes bid via the smart contract.

Step 3 (Compute the Allocation Scheme): The nodes run the auction algorithm off-chain based on the bid information recorded by the smart contract.

Step 4 (Offload Blocks): The losers offload blocks at the appropriate time. The winners continue to store blocks in response to data requests from losers and get paid.

B. Two-Stage Bidding Strategy

To prevent leakage of bid information during the auction process, we use two stages of bidding: 1) bid commitment and 2) bid disclosure. To ensure that the nodes' bidding information is nonrepudiation and the whole process is verifiable, we implement bidding on a smart contract.

Bid Commitment: Each node n_i calculates the commitment $com_i = Hash(b_i, height, str_i)$ and uploads it to the smart contract, where $Hash(\cdot)$ indicates a hash function, b_i indicates the node's bid, $height$ indicates the height of auctioned blocks, and str_i is a random string.

Bid Disclosure: Each node n_i uploads the actual bid $bid_i = (b_i, height, str_i)$ to the smart contract. All nodes are able to verify that bid_i and com_i match.

In order to save gas overhead, the smart contract maintains only two lists and does not include any computational processes. The nodes perform the hash calculation off-chain.

C. Fairness of Auction

With heterogeneous and limited resources of edge devices, it is necessary to achieve fair storage allocation, which is directly related to the stability of the system. We evaluate the fairness in two dimensions.

Fairness degree cost (FDC) describes the storage resource consumption: $\text{FDC}_i = s_i^u/(s_i^t - s_i^u)$, where s_i^u and s_i^t are used storage and total storage. FDC_i denotes the cost paid by a node to store blocks. Nodes with more storage resources pay less for storing blocks of the same size and have a higher priority in storage allocation.

Fairness degree advantage (FDA) describes the initiative of nodes to participate in the auction: $\text{FDA}_i = t_i \in [0, 1, \ldots, T]$,

where t_i is the number of times a node has participated in auctions since its last win. We encourage nodes to actively participate in the auction, and each failed bid increases the probability of success in the next one.

D. Second Price Auction

We design a second price auction with multidimensional information, taking into account the nodes' preferences and the system's fairness. Based on nodes' bids, each node is assigned to store those blocks that are most relevant to itself, reducing additional data requests. Based on fairness metrics, nodes are incentivized to participate in the auction and maximize the utilization of storage resources.

Ranking of Competitive Advantages: The winners will pay the second highest price. In our design, the nodes also have different fairness metrics, which should be included in the price paid by the winners. We map FDA, FDC, and bids into a 3-D space. We define the ideal value of auctioned blocks as the highest bid with the lowest cost and the highest advantage, i.e., the ideal point $P_{\text{ideal}}(0, T, \text{Bid}_{\max})$. The smaller the Euclidean distance d_i between the node's bid information $P_i(\text{FDC}_i, \text{FDA}_i, b_i)$ and the ideal point P_{ideal}, the greater the competitive advantage.

Data Redundancy Under Cost Tradeoffs: The degree of block redundancy (i.e., the number of winners) affects the storage cost and transmission overhead. The more winners, the higher the storage cost to the system. In contrast, the more losers, the higher the transmission overhead associated with the requested blocks. A tradeoff is made between the two to determine the level of redundancy.

The auction algorithm ranks the distances of all nodes and sequentially selects the nodes with a large competitive advantage to add to the winner queue until the sum of storage cost and transmission overhead is minimized. The details are as follows:

$$\min \sum_{i=1}^{N-M} \Delta\text{FDC}_i x_i + A \sum_{i=1}^{N-M} b_i(1-x_i) \tag{1}$$

$$\text{s.t. } \max|P_i P_{\text{ideal}}| \le \min|P_i P_{\text{ideal}}|(1-x_i) \tag{2}$$

$$x_i \in \{0, 1\}. \tag{3}$$

In the above formulation, x_i is the marker variable and $x_i = 1$ means that node i is selected as the winner. ΔFDC_i is the storage variation of the winner, which indicates the storage cost paid. The bid b_i indicates the importance of the block, which is proportional to the frequency of access. So the additional network transmission overhead of the system is represented by the sum of the bids of the losers $b_i(1-x_i)$. A is the scale factor that represents the tradeoff between storage overhead and transmission overhead.

The price paid by the winners is calculated according to the shortest distance of the losing side. We define that if there are two points that are equidistant from the ideal point, they have the same competitive advantage. When the fairness of two points is the same, their bids can only be directly compared. In Fig. 2, node x calculates the price it should pay based on node y, $d_x < d_y$. All meaningful points on a sphere with center P_{ideal} and radius d_y are equivalent to the point P_y. The

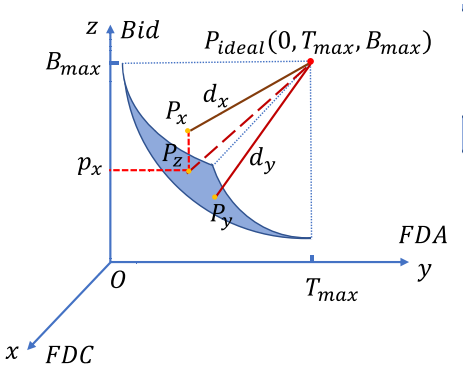

Fig. 2. Second price of multidimensional information.

projection P_z of the point P_x on the sphere, which has the same fairness as P_x. So the price paid by node x is $p_x = b_z$. In addition, $B_{x,y} = b_z - b_y$ is the fairness repayment of node x with respect to node y. The price paid by the winners was divided between all the losers. When two distances are equal, the node with the higher bid has higher priority.

We assume that the winning node i and the losing nodes k, j are ordered as follows:

$$\ldots \le d_i \le \ldots \le d_k \le \ldots \le d_j \le \ldots$$

where d_k is the shortest distance among the losing nodes. The price to be paid by node i is

$$p_i = \text{Bid}_{\max} - \sqrt{d_k^2 - \text{FDC}_i^2 - (T - \text{FDA}_i)^2} \le b_i. \tag{4}$$

All nodes obtain nonnegative utility, ensuring individual rationality and budgetary equilibrium. If node j wins the auction by raising its bid, it will pay

$$p_j = \text{Bid}_{\max} - \sqrt{d_k^2 - \text{FDC}_j^2 - \left(T - \text{FDA}_j\right)^2} > b_j. \tag{5}$$

Node j will receive a negative gain. If node j raises its bid but does not win the auction, the number of winning nodes may decrease according to (4), which will also make node j receive less gain. Therefore, the truthful bid is the optimal choice for the node.

E. Reputation Management

We design the reputation mechanism. There is no trust between nodes in the blockchain and reputation value is used to describe the trust cost. Reputation is not only a ticket for nodes to participate in the auction (requiring a nonnegative reputation value) but also a bargaining chip for bidding. A winning node stores a copy of the auctioned blocks, paying a reputation value as the trust price for the losing nodes to trust it. The higher the reputation value of the bid, the more important these blocks are to the node, and the higher the trustworthiness of the copies stored by the node. The reputation value is calculated as follows:

$$R_i = \begin{cases} \max\left(R_{\max}, R_i - p_i\right), & R_i \ge 0 \\ R_{\text{initial}}, & \text{initialized} \\ -R_{\max}, & \text{node } i \text{ does evil.} \end{cases} \tag{6}$$

Nodes are given an initial reputation value when they join the blockchain. In each auction round, all nodes participating in the auction should pay a reputation value p_i, which is negative for the losers. The reputation value has an upper limit $R_{\max}$. If there is a malicious node (winner) that does not store blocks and refuses to respond to data requests from other nodes, its reputation will be reduced to a minimum value $-R_{\max}$.

The reputation mechanism is implemented through smart contracts. Each reputation value update (auction settlement) is executed by a particular auctioneer and then each node participates in the verification. The decentralized nature of smart contracts makes reputation updates completely transparent and prevents private malicious tampering.

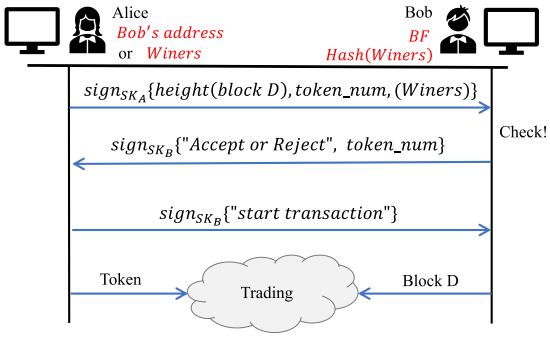

Fig. 3. Identification step. Bob is a winner of the block D. Alice asks Bob for block data. Bob charges different fees based on Alice's identity.

IV. Block Trading

The losers have ample incentive to offload the blocks because of the limited storage. We add the block buffer to optimize block access. To motivate the winners to provide storage services, we design a fair trading of block data to reward winners.

A. Reward and Punishment Mechanisms

Lazy winners do not store some relatively low-value winning blocks but rely on other winners to access this data. A lot of laziness will lead to data loss. Irrational malicious nodes do not even respond to requests from other nodes. Therefore, we design some reward and punishment mechanisms.

The key to implementing these mechanisms is to identify nodes (winners or losers). We describe the auction result AR as: $AR = [height, \vec{x}, \vec{p}, Hash(Winers), BF]$, where $\vec{x}$ and $\vec{p}$ are the allocation scheme and the price paid, $Hash(Winers)$ is the hash of the list of all winners' addresses, and BF is a binary vector into which the Bloom Filter maps the addresses of all winners [17]. Bloom Filters are widely used to quickly check if an element is in a set. Taking the address of each winning node as input, compute the values of some hash functions and set the corresponding position $Hash(address)$ mod $length$ to 1. Bloom filters suffer from the problem of false positive matches, i.e., for an element that does not exist in the set, the Bloom filter may think it does. When this occurs, additional evidence is needed.

1) Lazy Nodes: We design a soft punishment to discourage laziness. When two parties trade block data, the data provider verifies the identity of the requester and charges the winner (laziness) more than the loser. The identity of winners makes lazy nodes request auctioned blocks with higher costs, which drives them to actively play the role of winners to gain more benefits.

The identification process is shown in Fig. 3. When the nodes receive the auction result AR, the winner (Bob) saves $Hash(Winers)$ and BF. The loser (Alice) needs to determine whether there is a false positive match for herself, and if so, save the list $Winers$ as evidence. In the identification step, if Alice's request message is appended with the list $Winers$, Bob checks it by $Hash(Winers)$. If not, Bob determines Alice's identity based on BF.

2) Malicious Nodes: We define malicious behavior as the winning node rejecting a block trade for an incorrect reason. When malicious behavior occurs, the node uploads the accusation record and submits a deposit to the smart contract [18]. The accused node is voted on. If the accusation is not true, the plaintiff's deposit will be confiscated and paid to the participating nodes as the price of malicious defamation. If the accusation is true, the plaintiff will recover the deposit, and the defendant node needs to pay the same amount to be divided equally by the participating nodes. If the defendant is unable to pay, it is permanently removed from the system.

B. Fair Trading

In the case of mutual distrust, it is necessary for the winners and the losers to achieve a fair trading, i.e., both parties receive the correct block data and reward at the same time, and neither can deny it. Smart contracts and zero-knowledge proofs are widely used to achieve fair trading without trusted third parties [19], [20]. In difference, we use computationally more efficient asymmetric encryption to achieve this, since the losers used to have all the knowledge.

The process is shown in Fig. 4. Before Alice is ready to unload the block D, she sends a request to Bob asking if he is willing to provide storage services. If Bob refuses, he will be blacklisted by Alice, and Alice will ask other winners. If accepted, Bob computes a pair of public and private keys PK, SK and sends PK to Alice. Then, Alice encrypts the block D with the public key $D' = E(PK, D)$ and stores its hash value $HD' = Hash(D')$. Eventually, both the original data D and the encrypted data D' are deleted.

When Alice requests block D from Bob, Bob encrypts block D and sends $D'_b = E(PK, D)$ to Alice. Alice judges whether two hash values are equal $HD'? = Hash(D'_b)$. Subsequently, Alice publishes a smart contract with *publickeylock* PK and *timelock* Ti on the blockchain and places a deposit into the smart contract [21]. The lock can only be opened with a private key that matches the public key PK within a specified period of time Ti to obtain the deposit. Bob calls this smart contract and uses the private key SK as the input parameter to trigger the internal transfer logic and get the deposit. At the same time, this process is packaged as a transaction and broadcast on the entire network, Alice also knows the SK and completes the decryption $D = D_b = DE(SK, D'_b)$.

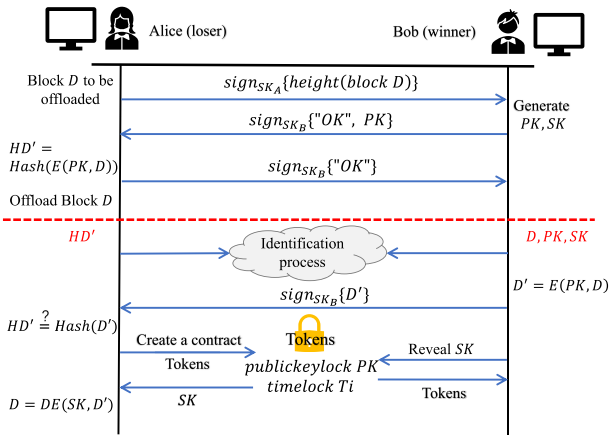

Fig. 4. Request step and Trade step. Use asymmetric encryption to achieve fair trading.

TABLE I
FACTORS AFFECTING AUCTION BENEFITS

Notation	Description
NN	Number of blockchain nodes
NB	Number of blocks
BS	Average block size (KB)
A	The scale factor in Eq.(3)
ε	The upper bound of false positive

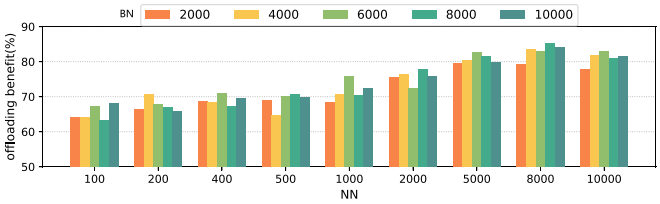

Fig. 5. Offloading benefits with different number of blocks (NB) and the number of nodes (NN). $BS = 52$, $A = 1$.

The suspension of the trading will not bring losses to both parties. Alice is able to judge the correctness of the encrypted block before publishing the smart contract. If the lock is not opened within time *Ti*, the deposit will be returned to Alice. Bob executes the smart contract locally, checking that its internal logic and deposit are correct. Bob will broadcast the transaction only after the check passes. Fair trading is achieved using the atomicity of blockchain operations and asymmetric cryptographic verification. Neither party can benefit from a broken promise.

V. EXPERIMENT

In this section, we evaluate our proposed auction algorithm and fair trading. We perform simulations on a computer with an AMD Ryzen ThreadRipper 3970X processor and 128 GB RAM. Docker[1] is an open-source application container engine that we use to simulate multiple distributed nodes. Each node runs the Go-Ethereum[2] blockchain framework in the container. We have implemented smart contracts for the auction algorithm using Solidity language, including user registration, reputation management, submission of bids, and algorithmic solution. In the case of a small number of users, smart contracts can be used to complete the entire auction process. But when a large number of users participate, it is not practical to complete the auction process with smart contracts due to the limitations of gas. Therefore, we adopt on-chain bidding information submission, off-chain auction algorithm calculation, and on-chain scheme result consensus. We simulate the off-chain auction algorithm with Python 3.9.

A. Storage Costs

We evaluate the auction algorithm from several perspectives. Using fully replicated storage as a baseline, the benefit of the auction is the percentage reduction in storage overhead after block offloading. The factors affecting the benefits are shown in Table I. The results of each experiment are repeated ten times and averaged.

Scalability: Fig. 5 shows the offloading benefits achieved by the auction algorithm for different number of nodes and different number of blocks. The offloading benefit hardly changes as the number of blocks increases, but it increases as the number of nodes increases. Intuitively, in a fair auction, more nodes can share the storage pressure of the system to some extent. In each auction round, on average 18%–35% of the nodes are selected to store the entire block, while the other nodes only store the block header. In addition, each node temporarily stores blocks in the cache and records the corresponding auction result information, including the winner's address and Bloom filter. This additional data occupies only a small amount of storage space. Overall, the auction algorithm ultimately achieves an average offload benefit of 65%–81% and better scalability in terms of number of nodes and number of blocks.

Additional Overhead: In our design, nodes need to store auction results (winning node addresses, hash values, boolean filters, etc.), which are the basis for fair trading. Depending on the specific number n of winners in each auction round, the length of the bloom filter $L \geq n \log_2 e \times \log_2(1/\varepsilon)$ [22]. Table II shows the additional storage space occupied by the auction results for different upper bound of false positive rate and number of nodes. A suitable ε can reduce the additional storage overhead to some extent. Inevitably, the percentage of additional storage overhead increases as the number of nodes increases. In total, nodes pay only a minimal storage cost (less than 0.76%) to store auction results.

Applicability: In different blockchain application scenarios, network resources and storage resources are not equivalent. We simulate the resource situation in different scenarios by tuning the scaling factor in (3). In addition, the block size varies widely in different scenarios. We randomly sample the size of a portion of Ethernet blocks in different time periods. Table III shows the offloading benefits achieved by the auction algorithm with different block sizes and different scale factors. In general, the block size is not the main factor affecting the

[1]https://www.docker.com/products/container-runtime
[2]https://ethereum.github.io/go-ethereum

TABLE II
PERCENTAGE OF ADDITIONAL STORAGE OVERHEAD (%)
$NB = 2000$, $BS = 52$, $A = 1$

NN \ ε	0.1	0.05	0.01	0.005	10^{-3}	10^{-4}	10^{-5}
100	0.04	0.03	0.03	0.03	0.03	0.04	0.05
200	0.06	0.05	0.04	0.04	0.05	0.06	0.07
400	0.1	0.07	0.06	0.07	0.06	0.08	0.1
500	0.12	0.08	0.06	0.07	0.08	0.11	0.12
1000	0.17	0.11	0.07	0.07	0.07	0.12	0.1
2000	0.32	0.19	0.11	0.11	0.12	0.14	0.17
5000	0.59	0.32	0.19	0.18	0.15	0.2	0.25
10000	0.76	0.69	0.29	0.31	0.25	0.43	0.32

TABLE III
OFFLOADING BENEFIT (%) $NN = 200$, $NB = 2000$, $\varepsilon = 10^{-3}$

A \ BS	16 Jun	17 Jun	18 Jun	19 Jun	20 Jun	21 Jun	22 Jun	AVG
	1.60	8.07	24.03	24.70	31.22	52.97	100.04	
0.1	99.86	99.96	99.98	99.97	99.98	99.98	99.98	99.96
0.2	99.84	99.97	99.98	99.98	99.98	99.98	99.99	99.96
0.5	99.36	99.77	99.86	99.86	99.84	99.89	99.89	99.78
0.6	99.12	99.7	99.77	99.75	99.12	99.8	99.8	99.58
0.8	85.53	85.68	88.01	94.39	82.17	83.85	90.75	87.20
1	64.76	64.11	65.68	65.9	67.05	62.22	67.3	65.29
1.2	47.46	50.99	55.02	49.8	52.14	48.42	54.04	51.12
1.6	32.15	33.33	33.42	33.81	31.31	32.57	32.66	32.75
2	21.63	23.39	24.42	24.9	23.5	25	23.89	23.82
5	4.56	7.73	7.99	8.1	8.18	8.06	8.92	7.65
10	1.03	4.35	4.95	4.97	4.96	5.22	5.27	4.39

auction algorithm. Instead, a smaller scale factor A means that storage resources are more scarce in that scenario, leading to a smaller number of winners chosen in the end. A bigger scale factor A means that network resources are more precious in that scenario, and more storage resources are sacrificed to avoid a large number of requests for blocks.

B. Fairness

In distributed scenarios, since devices are resource-limited and heterogeneous, achieving fair storage is important to fully utilize device resources. The above experiments show that different scaling factors make a large difference in the final realized offloading benefits compared to other influencing factors. So we evaluate the storage fairness row of the whole system with different scale factors. Fig. 6(a) shows the Gini coefficient[3] of FDC and average percentage of winners. When A is small, the winner is selected mainly from storage overhead considerations, resulting in a small number of winners. In particular, when A is less than 0.7, a large number of block auctions failed (with no winners), leading to an extremely uneven storage allocation among nodes and also destroying the integrity of the blockchain data. This situation should be strongly avoided. As A gets progressively larger, storage overhead and transmission overhead are considered together or more focused on transmission overhead. The number of winners tends to be stable and the Gini coefficient of the whole system is below 0.2, achieving good fairness. In practical applications, the scale factor can be dynamically selected according to specific application scenarios.

[3]The Gini coefficient is a commonly used index to measure the income gap of residents and is also used to measure fair storage in [23] and [24].

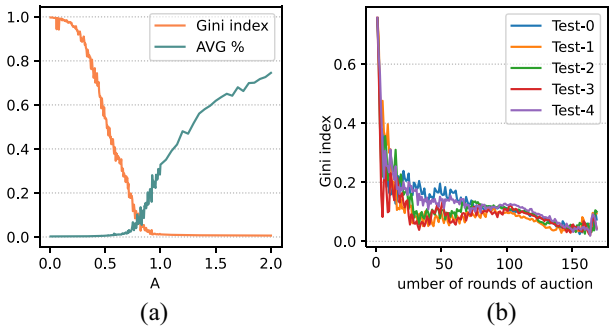

Fig. 6. Storage fairness of blockchain systems. (a) Gini coefficient of FDC and average percentage of winners. (b) Gini coefficient gradually stabilizes as the auction progresses.

In our design, the only way to gain reputation is to participate in auctions, which means that the reputation that each node has is relatively fixed throughout the auction process. Each node has a competitive advantage that matches its storage resources and selectively stores blocks by bidding freely. Therefore, as the auction proceeds, the Gini coefficient of the system will gradually decrease and eventually converge to a relatively stable value, as shown in Fig. 6(b).

VI. CONCLUSION

In this article, we propose a partial storage scheme to reduce the storage overhead of blockchain nodes. First, we design a second price auction model with multidimensional information that integrates the load balancing of the system and the nodes' preferences for block data. The losers offload blocks to reduce the storage overhead. The winners benefit by responding to requests for block data. Second, distributed nodes compute the auction solution off-chain and perform consensus on-chain. Then, we use smart contracts and asymmetric encryption to achieve a fair trading. Finally, the experiments demonstrate that the scheme has good scalability, applicability, and fairness.

REFERENCES

[1] X. Zhou, W. Liang, K. I.-K. Wang, and L. T. Yang, "Deep correlation mining based on hierarchical hybrid networks for heterogeneous big data recommendations," *IEEE Trans. Comput. Social Syst.*, vol. 8, no. 1, pp. 171–178, Feb. 2021. [Online]. Available: https://doi.org/10.1109/TCSS.2020.2987846

[2] X. Zhou, X. Yang, J. Ma, and K. I.-K. Wang, "Energy-efficient smart routing based on link correlation mining for wireless edge computing in IoT," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14988–14997, Aug. 2022. [Online]. Available: https://doi.org/10.1109/JIOT.2021.3077937

[3] X. Zhou et al., "Decentralized P2P federated learning for privacy-preserving and resilient mobile robotic systems," *IEEE Wireless Commun.*, vol. 30, no. 2, pp. 82–89, Apr. 2023. [Online]. Available: https://doi.org/10.1109/MWC.004.2200381

[4] S. Satija et al., "Blockene: A high-throughput blockchain over mobile devices," in *Proc. 14th USENIX Symp. Oper. Syst. Design Implement., (OSDI)*, 2020, pp. 567–582.

[5] C. Xu et al., "Making big data open in edges: A resource-efficient blockchain-based approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 4, pp. 870–882, Apr. 2019.

[6] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of BFT protocols," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 31–42.

[7] Y. Mao, S. Deb, S. B. Venkatakrishnan, S. Kannan, and K. Srinivasan, "Perigee: Efficient peer-to-peer network design for blockchains," in *Proc. ACM Symp. Principles Distrib. Comput.*, 2020, pp. 428–437.

[8] Y. Chen et al., "Forerunner: Constraint-based speculative transaction execution for Ethereum," in *Proc. 28th Symp. Oper. Syst. Principles*, 2021, pp. 570–587.
[9] S. Nakamoto. "Bitcoin: A peer-to-peer electronic cash system." 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[10] S. Han, Z. Xu, and L. Chen, "Jupiter: A blockchain platform for mobile devices," in *Proc. 34th IEEE Int. Conf. Data Eng.*, 2018, pp. 1649–1652.
[11] Z. Xu, S. Han, and L. Chen, "Cub, a consensus unit-based storage scheme for blockchain system," in *Proc. 34th IEEE Int. Conf. Data Eng.*, 2018, pp. 173–184.
[12] X. Qi, Z. Zhang, C. Jin, and A. Zhou, "BFT-store: Storage partition for permissioned blockchain via erasure coding," in *Proc. 36th IEEE Int. Conf. Data Eng.*, 2020, pp. 1926–1929.
[13] S. Wang et al., "ForkBase: An efficient storage engine for blockchain and forkable applications," *Proc. VLDB Endow.*, vol. 11, no. 10, pp. 1137–1150, 2018.
[14] X. Dai, J. Xiao, W. Yang, C. Wang, and H. Jin, "Jidar: A jigsaw-like data reduction approach without trust assumptions for bitcoin system," in *Proc. 39th IEEE Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 1317–1326.
[15] Y. Huang, J. Zhang, J. Duan, B. Xiao, F. Ye, and Y. Yang, "Resource allocation and consensus on edge blockchain in pervasive edge computing environments," in *Proc. 39th IEEE Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 1476–1486.
[16] J. Zhang, Y. Huang, F. Ye, and Y. Yang, "A novel proof-of-reputation consensus for storage allocation in edge blockchain systems," in *Proc. 29th IEEE/ACM Int. Symp. Qual. Service* 2021, pp. 1–10.
[17] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.
[18] H. Desai, M. Kantarcioglu, and L. Kagal, "A hybrid blockchain architecture for privacy-enabled and accountable auctions," in *Proc. IEEE Int. Conf. Blockchain*, 2019, pp. 34–43.
[19] I. Bentov and R. Kumaresan, "How to use bitcoin to design fair protocols," in *Proc. 34th Annu. Cryptol. Conf. Adv. Cryptol.*, 2014, pp. 421–439.
[20] G. Fuchsbauer, "WI is not enough: Zero-knowledge contingent (service) payments revisited," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2019, pp. 49–62.
[21] M. Herlihy, "Atomic cross-chain swaps," in *Proc. ACM Symp. Principles Distrib. Comput.*, 2018, pp. 245–254.
[22] Z. An, Q. Lin, L. Yang, and W. Lou, "Embracing tag collisions: Acquiring bloom filters across RFIDs in physical layer," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1531–1539.
[23] D. Wei, K. Zhu, and X. Wang, "Fairness-aware cooperative caching scheme for mobile social networks," in *Proc. IEEE Int. Conf. Commun.*, 2014, pp. 2484–2489.
[24] Y. Huang, X. Song, F. Ye, Y. Yang, and X. Li, "Fair and efficient caching algorithms and strategies for peer data sharing in pervasive edge computing environments," *IEEE Trans. Mobile Comput.*, vol. 19, no. 4, pp. 852–864, Apr. 2020.

Rui Pan received the B.S. degree in computer science and technology from Southwest Jiaotong University, Chengdu, China, in 2020. He is currently pursuing the Ph.D. degree in computer science and technology with Hunan University, Changsha, China.

His research interests include blockchain, game theory, and mobile-edge computing.

Yikun Hu received the Ph.D. degree from Hunan University, Changsha, China, in 2019.

His research interests includes parallel and distributed processing, cluster, and grid and cloud computing.

Chubo Liu (Member, IEEE) received the B.S. and Ph.D. degrees in computer science and technology from Hunan University, Changsha, China, in 2011 and 2016, respectively.

He is currently an Associate Professor of Computer Science and Technology with Hunan University. He has published over 40 papers in journals and conferences, such as the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE INTERNET OF THINGS JOURNAL, *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, *Theoretical Computer Science*, DAC, ICPADS, HPCC, and NPC. His research interests include game theory, approximation and randomized algorithms, and cloud and edge computing.

Dr. Liu won the Best Paper Award in IFIP NPC 2019 and the IEEE TCSC Early Career Researcher Award in 2019. He is a member of CCF.

Keqin Li (Fellow, IEEE) received the Ph.D. degree in computer science from the University of Houston, Houston, TX, USA, in 1990.

He is a SUNY Distinguished Professor of Computer Science with the State University of New York at New Paltz, New Paltz, NY, USA. He is also a National Distinguished Professor with Hunan University, Changsha, China. He has authored or coauthored nearly 800 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He holds more than 60 patents announced or authorized by the Chinese National Intellectual Property Administration. He has chaired many international conferences. His current research interests include cloud computing, fog computing and mobile-edge computing, energy-efficient computing and communication, embedded systems and cyber–physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent, and soft computing.

Mr. Li is among the world's top ten most influential scientists in distributed computing based on a composite indicator of Scopus citation database. He is currently an Associate Editor of the *ACM Computing Surveys* and *CCF Transactions on High Performance Computing*. He has served on the editorial boards for IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON SERVICES COMPUTING, and IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING.

Kenli Li (Senior Member, IEEE) received the Ph.D. degree in computer science from Huazhong University of Science and Technology, Wuhan, China, in 2003.

He was a Visiting Scholar with the University of Illinois at Urbana–Champaign, Champaign, IL, USA, from 2004 to 2005. He is currently a Full Professor of Computer Science and Technology with Hunan University, Changsha, China, where he is also the Dean with the College of Information Sciences and Engineering and the Director of National Supercomputing Center in Changsha. He has published more than 160 research papers in international conferences and journals, such as IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING, ICPP, and ICDCS. His major research interests include parallel computing, high-performance computing, and grid and cloud computing.

Prof. Li serves on the editorial board for the IEEE TRANSACTIONS ON COMPUTERS. He is an Outstanding Member of CCF.