Received 26 April 2020; | Revised 6 June 2016; | Accepted xxx

DOI: xxx/xxxx

RESEARCH

SmartStore: A Blockchain and Clustering Based Intelligent Edge Storage System with Fairness and Resilience†

Haiwen Chen[1,2] | Jiaping Yu[2] | Huan Zhou[2] | Tongqing Zhou[2] | Fang Liu[1,*] | Zhiping Cai[2,*]

[1]School of Design, Hunan University, Changsha, China

[2]College of Computer, National University of Defense Technology, Changsha, China

Correspondence

Fang Liu, School of Design, Hunan University, Changsha, Hunan, 410082, China. Email: fangl@hnu.edu.cn

Zhiping Cai, College of Computer, National University of Defense Technology, Changsha, Hunan, 410073, China. Email: zpcai@nudt.edu.cn

Present Address

College of Computer, National University of Defense Technology, Changsha, Hunan, 410073, China

Summary

With the development of edge computing, edge storage solutions are attracting widespread attention. When facing the requirements of lower latency and faster access speed from end devices, edge storage solutions are considered to be an alternative to the cloud. However, edges are usually owned by small organizations which have limited operations and maintenance capabilities. This makes these edge devices can be easily disabled by external attacks or internal hardware failures. Besides, the heterogeneity of the edge devices will also make it difficult to price the edge resources uniformly. To tackle these problems, we propose SmartStore: an auction mechanism based on blockchain to allocate edge resources. Considering centralized solutions have access bottlenecks and trust issues, we built SmartStore on the smart contract. With Bayesian game theory, SmartStore can analyze how Data Owners (DO) and Edges price the resources can maximize their benefits. From an economic perspective, both DOs and Edges can make full use of edge heterogeneous resources with SmartStore. Besides, a two-stage submission strategy is proposed to complete the sealed auction. Furthermore, considering the reliability of edge storage, we propose a cluster-based block distribution algorithm for SmartStore's intelligent edge recommendation process. SmartStore ensures the reliability of edge storage while maximizing the benefits and resource utilization of both parties. Finally, we conduct specific experiments on the proposed auction smart contract through "Ethereum" and the experimental results of implementation show the effectiveness and efficiency of our SmartStore.

KEYWORDS:

IoT devices, edge storage, sealed auction, blockchain, game theory

1 | INTRODUCTION

With the continuous development of the Internet of Things (IoT), the amount of data generated by varied kinds of sensors are increasing dramatically. According to Cisco's estimates, the amount of data generated by IoT, such as humans and machines, will be close to 850ZB by 2021[1]. Traditional cloud storage cannot meet the data storage requirements of low latency, high access, and fast backup. Edge computing[2] has emerged as a new computing paradigm for data computing and storage. With

†Haiwen Chen and Jiaping Yu contribute equally to the article.

edge computing solutions, computing power and data storage can be transferred from clouds to edges. Edge devices can be close to terminal devices geographically[3] to provide lower-latency computing and data access services.

Edge storage solutions, with the development of edge computing, have attracted widespread attention. Researchers have proved that portable or mobile devices, such as cameras or handheld tablets, with SD cards or other storage devices, can work as edge storage and send locally processed information to the cloud[1]. Due to the limitation of network bandwidth and storage capability, it is impossible to back up all the data of the smart cameras deployed at home or surveillance cameras in the community to the cloud. In the field of smart transportation(e.g., floating car sensors, videos, GPS information), Qiao et al.[4] use edge servers to store data, and adjust the distribution of data according to the state of the edge server through reinforcement learning-based intelligent scheduling, providing a new storage solution for intelligent transportation systems (ITS).

However, while edge storage brings convenience for data storage and access, it also brings some new problems. Storing data at the edge(e.g., SD cards, edge servers deployed by the third party) could be a security risk when facing attacks like network disconnection, power failure or equipment disruption. Without a well-established backup mechanism, data may be at risk of loss when abnormal cases happen to the edge. Some researchers recommend using erasure code[5] to perform redundant data backups on multiple edges to ensure reliable storage of data[3,6]. However, they didn't take into account the heterogeneity of edge devices and the requirements of storage reliability. Edges with different storage resources, such as bandwidth, access speed, and different storage spaces, can provide users with different quality of service. There has been limited work on distributed data storage on edges and fog resources[7,8,9,10,11], and most of those previous works are based on centralized resource allocation schemes, these schemes hindered the optimization of system performance and scalability. Besides, instead of improving the data storage reliability, most of these works focus on saving bandwidth or enhancing the speed of response. It is particularly important to provide a flexible and extensible backup scheme for edge storage data. It should fully utilize the edge heterogeneous resources to maximize the profit of multiple parties and the reliability of storage.

However, designing such a mechanism will face some challenges:

(1) How to maximize revenue? In edge storage, how to correctly and reasonably price storage resources so and maximize the revenue is an important issue. The edges tend to price storage space as high as possible to obtain greater revenue. But, overpriced storage space may not be accepted by the data owners. Therefore, how to make reasonable pricing in the system to maximize the revenue of the edges and achieve the purpose of encouraging consumers to use their storage space is a fundamental problem.

(2) Who does the resource allocation? When consumers request storage service, edges need to report their price and some attributes to a third-party platform, and a group of edges can be allocated to consumers through a reasonable algorithm. Choosing which platform to run the allocation algorithm is a tough problem. The client ends may not willing to provide their computing resources. If running on a third-party platform, it may have a single-point bottleneck problem. And the credibility issues such as conspiracy or unreasonably allocate storage requirements.

(3) How the data are distributed? The storage system needs to ensure the reliability of data storage. Simply considering the price factor may result in redundant data clustering in a geographic area. When some abnormal conditions occur (e.g., regional power outages or network disconnections, malicious persons damaging the storage SD cards of camera equipment), data aggregation in the same areas may lead to data loss. How to secure the reliability of storage based on realizing the benefit of both parties is also a key challenge.

To address these challenges, we designed a blockchain-based resource auction mechanism in the distributed edge storage scenario. The proposed auction mechanism considers not only price but also non-price attributes such as bandwidths and locations when determining edges. Using the smart contract of the blockchain as a trusted third-party platform can build a more reasonable and credible match between data owners and edges. Besides, we use an auction game theory based on Bayesian Nash equilibrium to analyze how to maximize the profit of two parties. Then, we implement the auction mechanism on the platform of Ethereum and verify the feasibility of auction games through simulation experiments. Our mechanism not only provides the price-based auction mechanism that considered both edge resources and reliability but also includes a blockchain to further enhance the system reliability. To the best of our knowledge, this is the first work that meets all the above requirements. Compared with some existing studies, the advantages of our work are shown in Table 1 .

In summary, the main contributions of this paper are as follows:

[1] https://www.micron.com/solutions/video-surveillance

TABLE 1 Comparison of related studies.

Studies	Decentralized	Fairness	Reliable	Scalable	Traceable	Price	Attribute
vStore[9]	High	Low	Low	Middle	-	-	-
He et al.[12]	Middle	Low	High	Middle	-	-	-
Linaje et al.[13]	Low	High	Middle	High	-	Involved	-
Xia et al.[3]	High	Low	Middle	Middle	-	Involved	Involved
Qiao et al.[4]	High	Low	Middle	Middle	-	-	Involved
Our Method	High	High	High	High	Yes	Involved	Involved

"-" means that it is not involved or discussed in the paper. Decentralized, Fairness, Reliable, and Scalable are features of the edge storage system. Traceable indicates whether the resource allocation can be traced. Price and Attribute refer to whether price and resource attributes are considered in resource allocation.

- We propose to use Bayesian Nash equilibrium, a game theory concept that has been widely applied in the field of economy, to analyze how edges and clients price the storage resources to maximize their revenue. By comprehensively weigh the benefits of buyers (Data Owners) and sellers (Edges), SmartStore can achieve reasonable pricing.
- We propose to apply smart contract as a trusted third-party platform to guarantee the credibility of multiple parties and implement a sealed auction mechanism that considers multiple attributes through a two-stage submission mechanism.
- We propose a cluster-based block distribution algorithm. It is not only a discrete blocks distribution algorithm but also a storage solution recommendation mechanism with multi-attribute priority and storage reliability for users.
- Our sealed auction implementation on the Ethereum platform verifies the feasibility and experimental evaluation shows the efficiency and flexibility of our SmartStore.

1.1 | Orgnization

We organized the rest of the paper as follows. Section 2 summarizes the related work. In Section 3, we briefly describe the system model. Section 4 analyzes the auction mechanism and the detail of our auction mechanism. Section 5 introduces our cluster-based selection algorithm. Secction 6 describes our specific application implementation and the performance. We conclude this article in Section 7.

2 | RELATED WORK

There are limited works on distributed data storage on edge and fog resources, as reviewed and classified in Moyasiadis, et al.[14]. We describe the decentralized storage solution from the perspectives of fog/edge computing[7,9,12,3,15]. FogStore[7] proposes a distributed key-value store on fog resources with replication and differential consistency. vStore[9] supports context-aware placement of data on fog and cloud resources, with mobile devices generating and consuming these data. It uses a rules engine to place and locate data based on its context metadata but ignores reliability as edge devices do not store data. He et al.[12] propose a data security storage model to improve the security of the storage in fog computing. Xia et al.[3] make the first attempt to formulate this Edge Data Distribution (EDD) problem as a constrained optimization problem from the app vendor's perspective to realize APP high-speed data access.

Aimed to achieve fair resource allocation, we use an auction mechanism to realize resource allocation and auction. This is a popular way to provide fair resource allocation between buyers and sellers in the case of competition[16]. In the study[17], the authors proposed a truthful auction mechanism in mobile cloud computing to achieve resource allocation between mobile devices and cloudlets. Kiani et al.[18] introduced a hierarchical mobile edge computing which contains different types of cloudlets and proposed a resource allocation mechanism with a two-time scale. Sun et al.[19] considered the industrial Internet of things scenario in which the edge node is a resource-rich data center and extended the above truthful auction mechanism. Bahreini et al.[20] solved the resource auction problem at the edge/cloud levels. However, it cannot meet the property of truthfulness. Peng

et al.[21] proposed the multiattribute-based auction toward resource allocation in vehicular fog computing. However, the above works either only consider the price factor of storage nodes when determining the winners or ignoring some factors like location, bandwidth, or consumer choice. In particular, none of the above methods are suitable for solving storage reliability problems in storage scenarios. Besides, some of these works cannot assure trustfulness, which is a problem solved by using blockchain technology in our research.

Besides, to maximize the profits of buyers and sellers, we use game theory[22] to analyze pricing strategies from an economic perspective. Li et al.[23] applied the incomplete information game theory and proposed a multi winners grid resource allocation model, which maximized social welfare and improved resource utilization. Pillai et al.[24] proposed a resource allocation mechanism for machines on the cloud. Guo et al.[25] modeled the bandwidth sharing problem as a Nash bargaining game, and proposed the allocation principles by defining a tunable base bandwidth for each virtual machine. In the research[26], a game-theoretic framework is proposed for resource allocation but the framework only considered a market model consisting of one INP and multiple MVNOS. Li et al.[27] introduced Bayesian Nash Equilibrium to realize multi-to-multi resource allocation based on resource attributes, which gave us a lot of inspiration. Some other studies[28,29,30] have proposed some resource allocation schemes based on game theory about energy constraints. However, all the above studies have only studied the attributes of resource allocation and cannot meet the requirements of simultaneously achieving storage reliability in our edge storage scenarios.

In this paper, we utilize a blockchain-based smart contract as a third party to realize the auction mechanism. Blockchain is considered to be a suitable platform for implementing incentive-driven distributed storage systems. In recent years, a line of work has been dedicated to the study of distributed storage systems for wireless networking applications based on public blockchains. Also, blockchain technology has also been introduced to solve problems such as data access control[31] and log recording[32]. In this article, blockchain can also be used to record data distribution related logs for resource allocation records. In the paper[33], a trading platform for Device-to-Device (D2D) computation offloading is proposed using a dedicated cryptocurrency network. Therein, resource offloading is executed between neighbor D2D nodes through a blockchain-based auction, and the block mining tasks are offloaded to the cloudlets. In Henning's research[34], a PoW-based public blockchain is adopted as the backbone of a P2P file storage market, where the privacy of different parties in a transaction is enhanced by the techniques such as ring signatures and one-time payment addresses. When identity verification is required for market access granting, e.g., in the scenarios of autonomous network slice brokering[35] and P2P electricity trading[36], the public blockchains can be adapted into consortium blockchains by introducing membership authorizing servers with little modification to the consensus protocols and smart contract design.

3 | SYSTEM MODEL

Figure 1 depicts the architecture of SmartStore mainly with five parts, the Data Owner (*DO*), Edge, Auction Crowd, Auction Smart Contract, and Winner. the Data Owner consists of diverse equipment (e.g., smartphones, tablet or personal PCs) with different storage requirements. Small and medium-sized data centers underutilized in enterprises/schools/hospitals and central telecommunication offices can serve as edges to provide storage service and get paid. Also, in the field of video surveillance, an SD card in a monitoring device can be used as an edge to provide storage space for other monitoring devices. When the *DO* releases a series of requirements, the eligible edges can join Auction Crowd for auction competition. They report their resources and prices to the smart contract for target resource indicators based on the details of the requirements released by the *DO*. The smart contract selects edge's resources and prices according to the requirements reported by *DO* and the edges are then selected in a second-round based on the *DO* bias. *DO* clusters the locations of edges that have passed the auction and then selects the appropriate edges in each cluster as the final winners.

Each DO may have a certain type of storage task and different requirements for different resources. For example, users with lower latency requirements may be willing to pay a higher price to choose faster edges. Assuming that each DO has the overall price for the target resources. Correspondingly, the resources of each edge contain different attributes, such as storage resource bandwidth, interrupted increment, and interrupted service online time. The prices of different resources are varied.

When a DO submits its request in a publicly accessible blockchain, it will trigger the generation of a smart contract for resource auction. The bidders meeting the requirements participate in the auction. Please note that the blockchain is responsible for verifying the availability of service. The edge will be punished if they cannot deliver the promised service. Various service auditing or data auditing methods can ensure that the service provided by edge works correctly.? After each edge makes a reasonable price, a group of edges whose prices meet the requirements is initially screened out by the smart contract. The price of

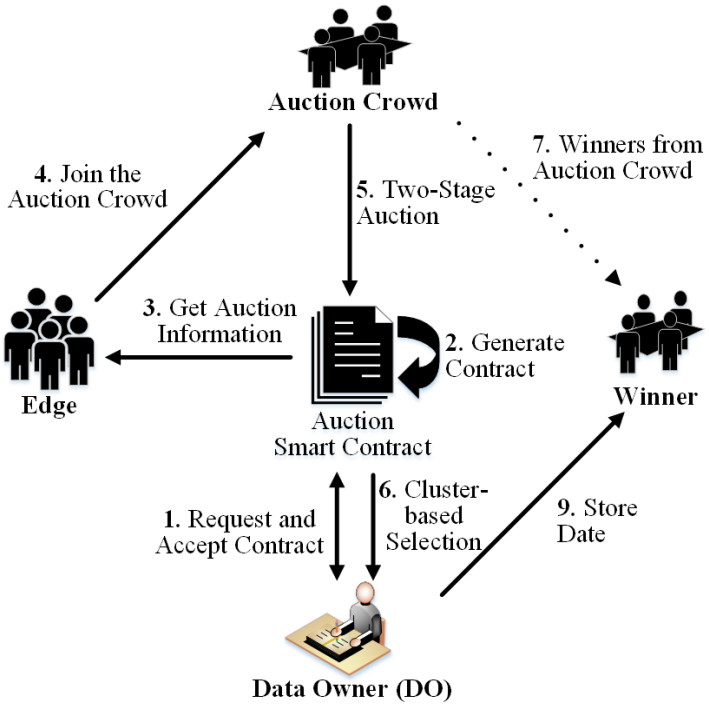


FIGURE 1 System overview of auction model based on smart contracts.

the edges and the price submitted by DO constitute a Bayesian game. The rationality of the auction and the solution to maximize the benefit is given in the paper.

After the DO obtains edges whose prices and resources satisfy the requirements suggested by the smart contract, edges are clustered using a geography-based clustering algorithm, and the number of clusters is equal to the number of blocks. Edges in each cluster are closest to the cluster centroid, which ensures that each edge selected is relatively independent with others selected from other clusters. In each cluster, an edge is selected according to a certain attribute and price requirement as the final winners, which ensures that the data are as dispersed as possible, and price or attribute requirements meet the requirements of DOs. We outline the edge selection process into the following steps:

(1) Initial. The client initiates the smart contract by submitting storage requirements. The submitted content includes a list of requirements for each resource $R = (R_1, ..., R_t)$, where t is the number of resources. The highest cost P_{total} required for storage is used to pay the edges that successfully bids. Besides, the smart contract also needs to include the two-stage submission time interval requirements, T_1 and T_2. Each edge needs to submit commitment c within the time of T_1 and the prices P, the secret random number s within the time of T_2.

(2) Bidding. Each edge and clients respectively price each resource P according to the pricing standard in Bayesian Nash equilibrium theory. Through the secret random number s generated by each node, combined with the pricing P, a commitment c is generated. Commitment c needs to be submitted within time T_1. All commitments submitted within the specified time are registered in the blockchain distributed ledger to ensure that the commitment cannot be modified.

(3) Revealing. Bidders and client submit the price P and secret random numbers s within the T_2 time interval as required, and the smart contract checks and calculates the commitment value $c^{'}$. The newly calculated commitment value $c^{'}$ is compared with the value c submitted by each node in the first stage to determine the validity of the submitted price. The valid price is stored in the smart blockchain.

(4) Reselecting. The client obtains the list of winning edges from the auction mechanism, including location information and related attributes. Based on the attributes and location information of the edges, clustering-based methods are used to select the final winning edges, taking into account the requirements of clients and the storage reliability.

In the Initial phase, Edges and DOs give reasonable prices for resources. Section 4.1 analyzes the price strategy, which guarantees that both DOs and Edges can get the maximum benefit, through Bayesian game theory and gives a detailed description

of the "bidding" and "revealing" process in Section 4.2. After the smart contract selects the edges, we use the clustering-based algorithm introduced in Section 5 to select the final edges that take into account DOs' requirements and storage reliability.

4 | SMART CONTRACT BASED AUCTION MECHANISM

In this section, we introduce the key technologies involved in the smart contract-based auction mechanism.

4.1 | Game Theory Based Price Strategy

In this section, we analyze the issue of resource prices from the economic perspective. Specifically, we target at determine a price that can maximize the interests of both parties. That is, how to set the price of the resource seller (marginal price) to make the price more acceptable to the resource demander (data owner). We use Bayesian games to analyze this problem.

In our scenario, there are two main parties involved in the auction: resource sellers (Edges) and resource buyers (Clients of Data Owners). For edges, they hope to sell as high as possible and selling their own resources as much as possible. The client hopes to buy the resources he/she needs at the lowest possible price. For both parties, it is hoped to obtain the highest expected benefits from the transaction according to their different resources. Besides, we assume that both parties to the auction have an accepted price that is not known to other participants. Moreover, all parties have a distribution function to measure resource prices. When the auction starts, all parties will price the resources they need or they own and each party submits a sealed price set of resources to the smart contract to form a static game.

We assumpt that m edges provide storage resources for n clients (Data Owners). P_{max} and P_{min} are the highest and lowest unit prices of resources in the market, $P_{max} = \{p_{max_1}, ..., p_{max_t}, ..., p_{max_k}\}$, $P_{min} = \{p_{min_1}, ..., p_{min_t}, ..., p_{min_k}\}$, $0 < j \leq k$, k is the type of resource. Besides, we assume that each node knows its own linear valuation function of resources. Then, We model the auction price problem as a game Γ of incomplete infomation and the auction game is defined as follows:

Definition 1. Game Γ: It is a $(m \oplus n)$-players game represented as a triple $(\mathcal{P}, \Sigma, \Pi)$.

- $\mathcal{P}$ means the set of m edges (sellers) and n clients (buyers), including the resources set that they own or require, $\mathcal{P} = \mathcal{C} \cap \mathcal{E}$, $\mathcal{C}$ and $\mathcal{E}$ are defined as follows.
  - $\mathcal{C} = \{c_1, c_2, ..., c_n\}$ $(n > 1, n \in \mathbb{N}^+)$ is the set of clients (buyers). R_i^C is the resources needed by client i, $R_i^C = \{r_{i1}^c, ..., r_{it}^c, ..., r_{ik}^c\}$.
  - $\mathcal{E} = \{e_1, e_2, ..., e_m\}(m > 1, m \in \mathbb{N}^+)$ is the set of m edges (sellers) and $R_j^E = \{r_{j1}^E, ..., r_{jt}^E, ..., r_{jk}^E\}$ represents the resource set that edge e_j provides to the system.
- $\Sigma = \Sigma_1 \cap \Sigma_2$ is the auction price strategy of all participates. We denote Σ_1 as the expected price strategy of clients and Σ_2 as the auction price strategy of edges.
  - In Σ_1, the client i's valuations for the k kinds of resources, $PV_i^C = \{pv_{i1}^c, ..., pv_{it}^c, ..., pv_{ik}^c\}$. pv_{it}^c is independent and identically distributed on $[p_{min_t}, p_{max_t}]$. For each resource in R_i^C, the price set that the client can accept is $P_i^C = \{p_{i1}^c, ..., p_{it}^c, ..., p_{ik}^c\}$.
  - in Σ_2, the edge j's valuations for the k kinds of resources, $PV_j^E = \{pv_{j1}^E, ..., pv_{jt}^E, ..., pv_{jk}^E\}$. pv_{jt}^E is independent and identically distributed on $[p_{min_t}, p_{max_t}]$. For each resource in R_j^E, the price that the edge j's bid price set is $P_j^E = \{p_{j1}^E, ..., p_{jt}^E, ..., p_{jk}^E\}$.
- $\Pi = \Pi^1 \cap \Pi^2$ is the set of payoff functions of two parties. where $\Pi : \Sigma \rightarrow Profit$ is the payoff function under a certain strategy.

The quantity set of the resources when edges j and client i reach a deal is $Q = \{q_1, ..., q_t, ..., q_k\}$, $0 < t \leq k$. From the Definition 1 , we get the total bid price of edge j for the resource set R_j^E is $\sum_{t=1}^k q_t p_{jt}^E$ and the highest acceptable price for client i is $\sum_{t=1}^k q_t p_{it}^c$. Besides, the client i evaluates the t-th resource as pv_{it}^c, and the price submitted to the smart contract is p_{it}^c, obviously, the client only benefits when $p_{it}^c < pv_{it}^c$ is satisfied; Similarly, the edge j evaluates the t-th resource as pv_{it}^E and the auction price is p_{it}^E, $p_{it}^E > pv_{it}^E$. All participants only know their own valuations, do not know the valuations of others, but everyone knows the valuation is uniformly distributed on $[p_{min}, p_{max}]$.

Definition 2. Price Strategy: Assume the pricing strategies of both parties are linear functions of their own valuations. That is, using linear pricing strategies:

$$p_{it}^C = a_{it}^C + b_{it}^C pv_{it}^C, \quad b_{it}^C > 0 \tag{1}$$

$$p_{jt}^E = a_{jt}^E + b_{jt}^E pv_{jt}^E, \quad b_{it}^E > 0 \tag{2}$$

In equations 1 The acceptable price strategy for any client i is a functions $\{p_{i1}^C, \cdots, p_{it}^C, \cdots, p_{ik}^C\}$ that specify the ask price of the client under each possible valuation. Similarly, the bidding strategy for any edge j is a collection of functions $\{p_{j1}^E, \cdots, p_{jt}^E, \cdots, p_{jk}^E\}$ that specify the bid price under different valuation scenarios.

Definition 3. Profits: If the client i and the edge j reach a deal, the total price will be:

$$D_{ij} = \frac{P_i^C + P_j^E}{2} Q = \sum_{t=1}^{k} \frac{p_{it}^c + p_{jt}^E}{2} q_t \,, \tag{3}$$

and the profits for the client i and the edge j are respectively:

$$\Pi^1(P_i^C) \,:\, \sum_{t=1}^{k} [pv_{it}^c - \frac{p_{it}^c + p_{jt}^E}{2}] q_t \,, \tag{4}$$

$$\Pi^2(P_j^E) \,:\, \sum_{t=1}^{k} [\frac{p_{it}^c + p_{jt}^E}{2} - pv_{jt}^c] q_t \,. \tag{5}$$

Besides, we denote $\Pi^1(P_{-i}^C)$ as the profits of all client participates without the client i's profit, $\Pi^2(P_{-j}^E)$ as the profits of all edge participates without the edge j's profit. Therefore, $\Pi(P_i^C, P_{-i}^C)$ or $\Pi(P_j^E, P_{-j}^E)$ is the profits of all participates.

Definition 4. Nash equilibrium A price strategy profile $\Sigma = \Sigma_1^* \cap \Sigma_2^* = \{P_1^{C*}, ..., P_n^{C*}, P_1^{E*}, ..., P_m^{E*}\}$ is the Nash equilibrium of game Γ if no clients or edges can further increase its profits by unilaterally changing its strategy at equilibrium. Mathematically,

$$\Pi(P_i^{C*}, P_{-i}^{C*}) \geq \Pi(P_i^C, P_{-i}^{C*}),\ 1 \leq i \leq n, \tag{6}$$

$$\Pi(P_j^{E*}, P_{-j}^{E*}) \geq \Pi(P_j^E, P_{-j}^{E*}),\ 1 \leq j \leq m. \tag{7}$$

If the game state reaches the Nash equilibrium, for the acceptable price of any client i and the bidding price of any edge j, no other higher prices can achieve higher benefits. This is a game of incomplete information. Through Harsanyi transformation[37], the game under the condition of incomplete information can be converted into a complete but imperfect information game. We introduce a virtual participant "Nature", which gives the corresponding of the valuations pv_{it}^C and pv_{jt}^E.

Theorem 1. If the game state reach the Bayesian Nash equilibrium, the each acceptable resource price P_i^C of any client i and the each bidding resource price P_j^E of any edge j should satisfy the following conditions:

$$\begin{aligned} &(a)\ \max_{p_{it}^C} \{pv_{it}^C - \frac{p_{it}^C + Exp^E}{2}\} * Prob^C,\ \ (b)\ \max_{p_{jt}^E} \{\frac{p_{jt}^E + Exp^C}{2} - pv_{jt}^E\} * Prob^E, \\ &\qquad\qquad s.t.\ \ max\Pi(P_i^C)\ \ and\ \ max\Pi(P_j^E) \end{aligned} \tag{8}$$

where pv_{it}^C and pv_{jt}^E within the interval $[p_{min_t}, p_{max_t}]$. $Prob^C = p\{p_{it}^C \geq p_{jt}^E(v_{jt}^E)\}$, $Exp^E = E[p_{jt}^E(pv_{jt}^E) | p_{it}^C \geq p_{jt}^E(pv_{jt}^E)]$, $Prob^E = p\{p_{it}^C(pv_{it}^C) \geq p_{jt}^E)\}$, $Exp^C = E[p_{it}^C(pv_{it}^C) | p_{it}^C(pv_{it}^C) \geq p_{jt}^E]$.

In equation (8a), the first item in the curly braces is the actual transaction price of the t-th resource when the i-th client and the j-th edge reach a deal; Exp^E is the expected value of the j-th edge's bid price when the transaction is successful; And $Prob^E$ is the probability that a transaction will succeed when the j-th edge is bidding p_{jt}^E.

In equation (8b), the second item in the curly braces is the actual transaction price of the t-th resource when the i-th client and the j-th edge reach a deal; Exp^C is the expected value of the i-th client ask price when reaching a deal; And $Prob^C$ is the probability that a transaction will succeed when the i-th client is asking p_{it}^C.

We reference the similar proof work in the literature[27] to resolve the problem of how to price the storage resources to achieve Nash equilibrium.

Proof. Considering that pv_{it}^C and pv_{jt}^E in the above assumptions are uniformly distributed on $[p_{min_t}, p_{max_t}]$, the probabilities and mathematical expectations for equations (8a) and (8b) can be calculated as follows:

$$\begin{aligned} Prob^E &= p\{p_{it}^C(pv_{it}^C) \geq p_{jt}^E\} = p\{(a_{it}^C + b_{it}^C pv_{it}^C) \geq p_{jt}^E\} = p\{pv_{it}^C \geq \frac{p_{jt}^E - a_{it}^C}{b_{it}^C}\} \\ &= \frac{p_{max_t} - \frac{p_{jt}^E - a_{it}^C}{b_{it}^C}}{p_{max_t} - p_{min_t}} = \frac{p_{max_t} b_{it}^C + a_{it}^C - p_{jt}^E}{b_{it}^C (p_{max_t} - p_{min_t})} \end{aligned} \tag{9}$$

Similarly,

$$Prob^C = p\{p_{it}^C \geq p_{jt}^E(pv_{jt}^E)\} = \frac{p_{it}^C - p_{min_t} b_{jt}^E - a_{jt}^E}{b_{jt}^E (p_{max_t} - p_{min_t})} \tag{10}$$

Besides,

$$\begin{aligned} Exp^C &= E[p_{it}^C(pv_{it}^C) | p_{it}^C(pv_{it}^C) \geq p_{jt}^E] = E[(a_{it}^C + b_{it}^C pv_{it}^C) | (pv_{it}^C \geq \frac{p_{jt}^E - a_{it}^C}{b_{it}^C})] \\ &= \frac{E[a_{it}^C + b_{it}^C pv_{it}^C, pv_{it}^C \geq \frac{p_{jt}^E - a_{it}^C}{b_{it}^C}]}{p\{pv_{it}^C \geq \frac{p_{jt}^E - a_{it}^C}{b_{it}^C}\}} = \frac{p_{jt}^E + a_{it}^C + b_{it}^C p_{max_t}}{2} \end{aligned} \tag{11}$$

Similarly,

$$Exp^E = E[p_{jt}^E(pv_{jt}^E) | p_{jt}^E(pv_{jt}^E) \geq p_{it}^C] = \frac{p_{it}^C + a_{jt}^E + b_{jt}^E p_{min_t}}{2} \tag{12}$$

Then, we can get the following translations:

$$Max_{jt}^E : \max_{p_{jt}^E} \{\frac{p_{jt}^E + \frac{p_{jt}^E + a_{it}^C + b_{it}^C p_{max_t}}{2}}{2} - pv_{jt}^E\} \frac{p_{max_t} b_{it}^C + a_{it}^C - p_{jt}^E}{b_{it}^C (p_{max_t} - p_{min_t})} \tag{13}$$

$$Max_{it}^C : \max_{p_{it}^C} \{pv_{it}^C - \frac{p_{it}^C + \frac{p_{it}^C + a_{jt}^E + b_{jt}^E p_{min_t}}{2}}{2}\} \frac{p_{it}^C - p_{min_t} b_{jt}^E - a_{jt}^E}{b_{jt}^E (p_{max_t} - p_{min_t})} \tag{14}$$

Considering that the general way to find the optimal solution is to make the first order partial derivative of equation equal to 0, and then verifying that the second order partial derivative of the equation is less than 0, the optimal solution of the equation can be obtained. Therefore, we first let the first order partial derivative of equations (13) and (14) equal to 0 respectively:

$$\frac{\partial(Max_{jt}^E)}{\partial(p_{it}^C)} = 0 \rightarrow p_{jt}^E = \frac{1}{3}(a_{it}^C + b_{it}^C p_{max_t}) + \frac{2}{3} pv_{jt}^E \tag{15}$$

$$\frac{\partial(Max_{it}^C)}{\partial(p_{jt}^E)} = 0 \rightarrow p_{it}^C = \frac{1}{3}(a_{jt}^E + b_{jt}^E p_{min_t}) + \frac{2}{3} pv_{it}^C \tag{16}$$

Besides, $\frac{\partial^2(Max_{jt}^E)}{\partial(p_{it}^C)^2} < 0$ and $\frac{\partial^2(Max_{it}^C)}{\partial(p_{jt}^E)^2} < 0$, so equations (15) and (16) are the optimal solutions. Comparing with equations (1) and (2), we can get p_{jt}^E and p_{it}^C as follows:

$$p_{jt}^E = \frac{1}{12} p_{min_t} + \frac{1}{4} p_{max_t} + \frac{2}{3} pv_{jt}^E, \;\; p_{it}^C = \frac{1}{4} p_{min_t} + \frac{1}{12} p_{max_t} + \frac{2}{3} pv_{it}^C \tag{17}$$

□

It can be seen that when clients and edges set prices of resources according to equation (17). Then, the game state can gain equilibrium. The pricing strategies for all participants are functions of their respective valuations, the highest and lowest market prices.

4.2 | Two-Stage Submission Auction

Since the transaction is transparent in the public blockchain, how to realize a sealed auction using a smart contract is a challenge. To ensure normal payment to the auction winner, DOs should transfer the highest price they can accept and some requirements (the number of blocks, bandwidth, throughput, etc.) to the smart contract during the bidding stage. Besides, the price stored in the blockchain may also be read by other bidders. This will cause bidders to change their bids accordingly instead of bidding truthfully, which will result in loss of profits for the seller and loss of fairness in the auction. For example, suppose that for a user with storage requirements, the maximum acceptable file storage value is \$2, once the bidder knows that the minimum required price is \$2, the bid is reduced to \$1.99 to win the auction, and for the bidder who bid first will bid \$1.99, and the bidder who bid later can bid \$1.98 to win the bid. Therefore, it is necessary and challenging to realize a sealed auction through blockchain.

To achieve a sealed auction and solve the fairness problem, we propose a two-stage submission mechanism to realize the "bidding-revealing" process. The proposal requires that the commitment be submitted first in the stage of "bidding", and all bidders will uniformly reveal the price corresponding to the commitment after all commitments are submitted. Edge's conditions (bandwidth, throughput, etc.) submitted by each bidder do not need to be sealed, and these conditions can be detected by the system. If a bidder reports false positive information, it will be punished. Submit the commitment first to protect the quotation and prevent bidders from modifying their quotations by reading the prices of other bidders. We use the Keccak-256 hash function[38] to realize the conversion from price to promise.

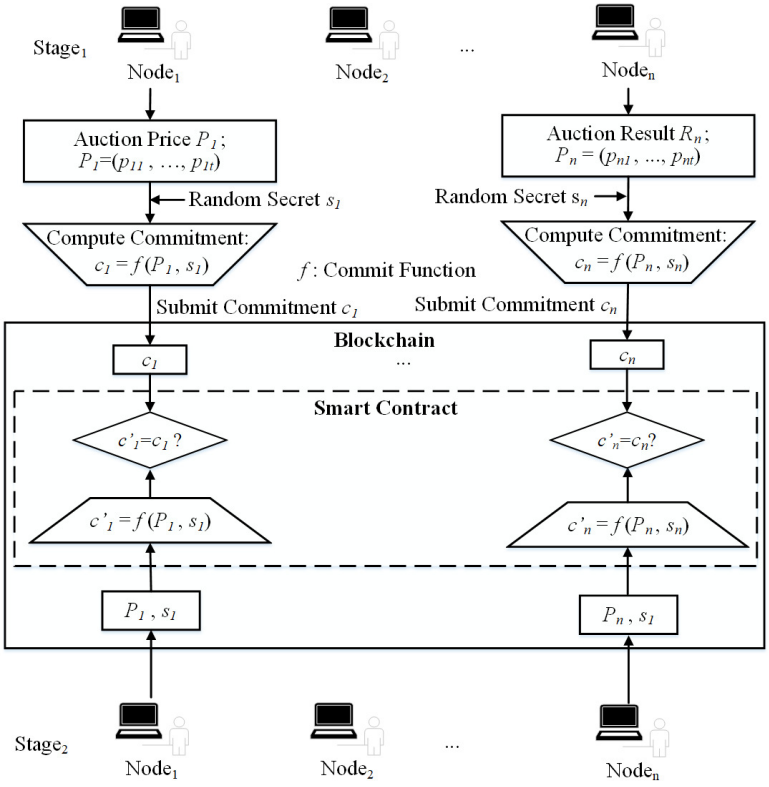


FIGURE 2 The two-stage submission mechanism.

In the first stage of the bidding process, to ensure the fairness of the auction of each node, both the edges and the clients need to submit a commitment c. Commitment c at the second submit stage can reveal the auction price of storage services and the acceptable price of the client. We assume that the resource price is $P = (p_1, \cdots, p_i, \cdots, p_n)$, and p_i represents the price of different types of resources, such as bandwidth, throughput, etc. Each node first needs to submit its own commitment c to the smart contract based on its auction price. Commitment c is generated by each node according to the price P and a secret random number s, by calculating the commitment $c = Comment(P, s)$. Each node has a random number s, so the commitments submitted by each node are not the same. If a collision is promised during the submission stage, then the submitted node will

regenerate the secret random number s and recalculate c for submission. Using commitment, each node submits promises within a certain period to ensure that the price will not change. The node that does not submit a promise is deemed to have given up the current round of auction.

Once the bidding time is over or all nodes have submitted their commitment. The commitment c is registered in the blockchain, and the non-tamperable property of the blockchain ensures that the commitment cannot be modified. Next, the bidder sends the auction price P and secret random number s to the smart contract in the revealing stage. After all nodes submit the price and secret random number or the submit time is over, the smart contract calculates c according to the secret random number s and the action price P through the $Commit$ function. We denote the commitment calculate by smart contract is $c^{'}$, $c^{'} = Commit(P, s)$. If $c^{'}$ is equal to c, it means that the bidder has complied with the bidding rules and submitted the quotation at the time of promise honestly, then the bid price will be registered in the blockchain. $c^{'} != c$ means the bidder modifies the price during the price submission stage, and the bid is invalid.

5 | CLUSTER-BASED SELECTION ALGORITHM

In some studies of computing resource allocation, most of the goals pursued are maximization of benefits. However, in storage scenarios, the reliability of file storage is an indicator that cannot be ignored. In the edge computing scenario, data are distributed to edge servers after splitting into blocks. The scattered and redundant storage of blocks can ensure that the data can always be recovered after one or more blocks are lost. However, when we seek to maximize revenue through the auction, we ignore the issue of data storage reliability. Therefore, we propose to use a cluster-based selection algorithm to further filter edges, achieving the two goals of maximizing revenue and high file reliability.

5.1 | Analysis of block distribution problems

The files stored are fragmented to many blocks to be stored in different edges. Some edges may be owned by the same organization or communities that have the same public power supply settings, access gateway equipment, and the same external threats. Suppose that the blocks of a file are gathering in a concentrated area, and the above-mentioned accidents, such as power outages, network disconnections, and attackers' vandalism have occurred in this geographic area. This will directly cause the blocks in this geographic area to be offline at the same time, and the data will be lost. Besides, although it is likely to be true that not all the aggregated edges belong to the same organization, geographical clustering can also cause these edges to face the same threat. For example, the memory cards on video surveillance equipment gathered in a certain geographic area may store the data of this node or other nodes in the edge storage system. When some nodes in this geographic area are maliciously damaged, it may cause data loss. Therefore, by distributing blocks in different geographic locations of the storage system (specified ranges according to system requirements or data transmission delay), ensuring that all blocks of a unified file are non-gathering and dispersed as much as possible can achieve higher file storage reliability.

We randomly generate data and use the form of Figure 3 to further illustrate the problem we described above. The locations of all points in Figure 3 represents locations of edges in storage services in different campuses, hospitals, and other organizations within a geographic range. From the above figure, we can intuitively observe that the edges in the two circles are very clustered. If too many data blocks of the same file are concentrated in the circled position, and internal or external threats occur at this position, the file will face a greater risk of loss.

5.2 | Cluster-based Block Distribution Mechanism

In response to the problems described above, this paper proposes a storage configuration recommendation that uses a cluster-based block distribution mechanism to achieve multi-objective selection.

Supposed that m Edges win the auction and the Client can obtain the price information and location information of the winners $(e_1, e_2, ..., e_m)$. Then, the client performs central clustering according to the location information of the edges. The number of clusters is the same according to the number of blocks. For example, if there are k blocks, the number of clusters is k. After clustering, all edges fall into k relatively independent spaces. The edges transmitted from the k spaces constitute a combination to achieve as much price or resource demand as possible based on considering reliability.

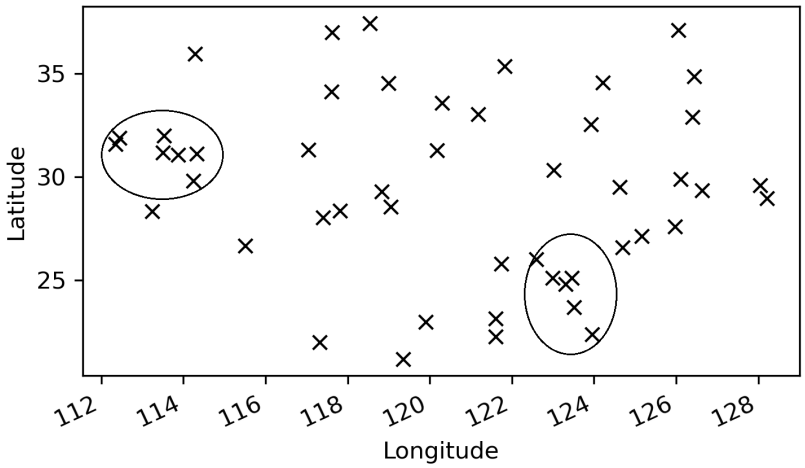


FIGURE 3 Example of location distribution.

Since the total number of edges and the blocks are known, we choose to use KMeans central clustering algorithm for the clustering process. The KMeans algorithm is the most commonly used clustering algorithm. In our paper, the known set to be clustered is $E = \{e_1, e_2, ..., e_m\}$, the number of subspaces that need to be clustered is k, therefore, the problem of clustering is to find k locations $L = \{l_i\}_{i=1}^{k}$ to minimize the following cost function:

$$cost(E, L) := \sum_{e \in E} \min_{l_i \in L} ||e - l_i||^2.$$

Equivalently, the KMeans clustering problem can be considered as the problem of finding the partition $P = \{p_1, p_2, ..., p_k\}$ of E into k clusters that minimizes the following cost function:

$$cost(P) := \sum_{i=1}^{k} \min_{l_i \in \mathbb{R}^{2d}} \sum_{p \in P} ||p - l_i||^2.$$

KMeans clustering is used to find k clusters, and the set of all positions is equally divided into k clusters. The k relatively independent clusters ensure that the sum of the distances between each position point and the centroid is the smallest. The client can select edges according to the attributes required by itself. For example, if bandwidth is required first, then each cluster can be sorted according to the bandwidth, so as to select the best edges in each cluster. However, if the resources in some clusters are consistent or there are multiple consistent optimal, how to give an edge selection is a problem. If some edges have been selected in some clusters, how to continue selecting from the remaining clusters.

We make supplementary modifications to KMeans algorithm. After the clustering and sorting by the attributes of one or more items in each clusters, some edges in clusters can be selected, while there are multiple edges in some other clusters. At this point, we select edges based on the principle of storage reliability. Supposed that the selected edges are $E = \{e_1, e_2, ..., e_t\}$ located in the cluster $P = \{p_1, p_2, ..., p_t\}$, the centroid set is $CP = \{cp_1, ..., cp_t\}$, the unselected cluster $\hat{P} = \{p_{t+1}, p_{t+2}, ..., p_k\}$, and the centroid set of unselected cluster is $\hat{CP} = \{cp_{t+1}, ..., cp_k\}$. We continue to iterate the following process to select the optimal edges in each of the remaining clusters.

$$\forall p \in \hat{P} : \max_{l_j \in p}(\min_{cp_i \in CP} |cp_i - l_j|).$$

By maximizing $cost(\hat{P})$, we find the edges farthest from the selected set in the remaining clusters one by one. Of course, this is not necessarily the optimal choice, because the above selection method is related to the order of the remaining clusters. If we want to find the optimal combination, we can constantly change the iterative sequence of the remaining clusters. Considering that the iterative selection process is located on the DO's side, the complexity cost is too high. Therefore, we only realize the selection by randomly selecting the remaining clusters. In the following experiments, we will show through a large number of experiments that the effect of the above-mentioned maximization of $cost(\hat{P})$ selection is far greater than the method of random selection in each cluster.

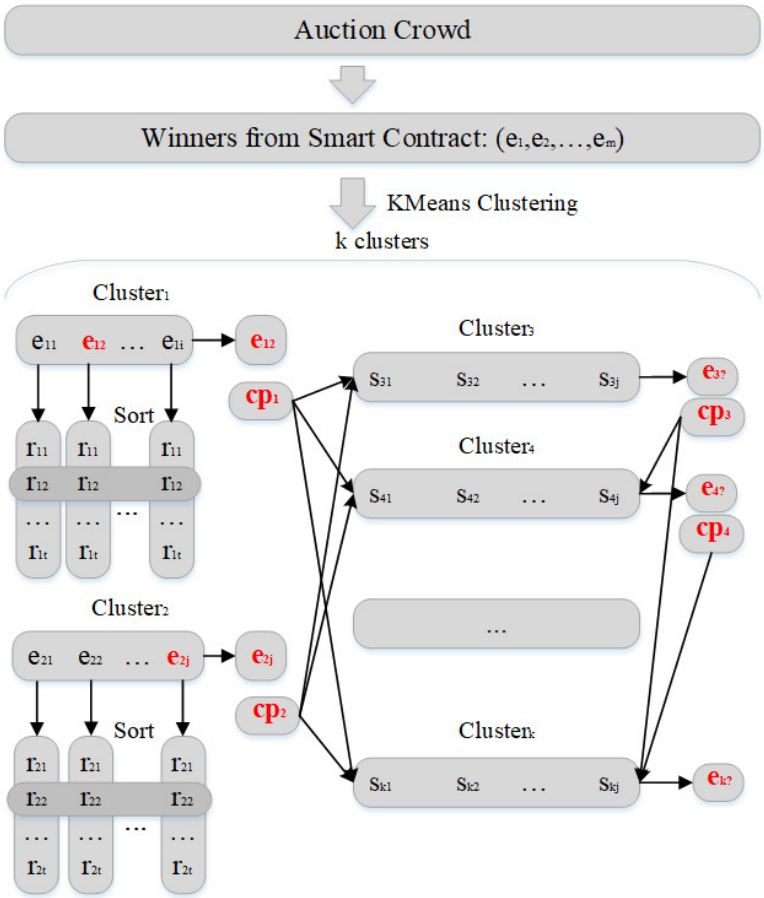


FIGURE 4 Clustering based selection mechanism.

The specific selection process is shown in Figure 4 , the winners $(e_1, e_2, ..., e_m)$ from smart contract are clustered into k clusters $(cluster_1, ..., cluster_k)$. We assume that edges e_{12} and e_{2j} have been selected according to a certain attribute in cluster $Cluster_1$ and $Cluster_2$, and the centroids of $Cluster_1$ and $Cluster_2$ are cp_1 and cp_2. Then, we can find the farthest edge $e_{3?}$ from cp_1 and cp_2 in $Cluster_3$ and find the farthest edge $e_{4?}$ from cp_1, cp_2, and cp_3 in cluster $Cluster_4$. Continuously iterating the above process, we finally choose the edges $(e_{12}, e_{2i}, e_{3?}, e_{4?}, ..., e_{k?c})$ that takes into account storage reliability. We give our pseudo-code implementation in Algorithm 1.

6 | PROTOTYPE IMPLEMENTATION AND EVALUATION

There are mainly two parts in the implementation and evaluation process: one is the smart contract implementation and measurements on the consumption of interfaces; The other is the analysis and experiment of the cluster-based data distribution algorithm.

6.1 | Auction Smart Contract Implementation and Analysis

6.1.1 | Auction Smart Contract Implementation

We implement an auction model based on smart contracts in Ethereum. Figure 5 shows the state transition process in the smart contract. Three roles are mainly involved in the model: Client(C), Service Node(SN), and Service Provider(P). We define the states in the smart contract: “Init”, “Active”, “Bidding”, “Reveal”, and “Complete”. The interfaces used for the transition between states in the figure belong to the content of the smart contract, “{Role}: {Interface Name}”. The judgment box is used

for selection when the contract conflicts. In Ethereum, a specific role can access a specific breach to realize the interaction between the role and the contract.

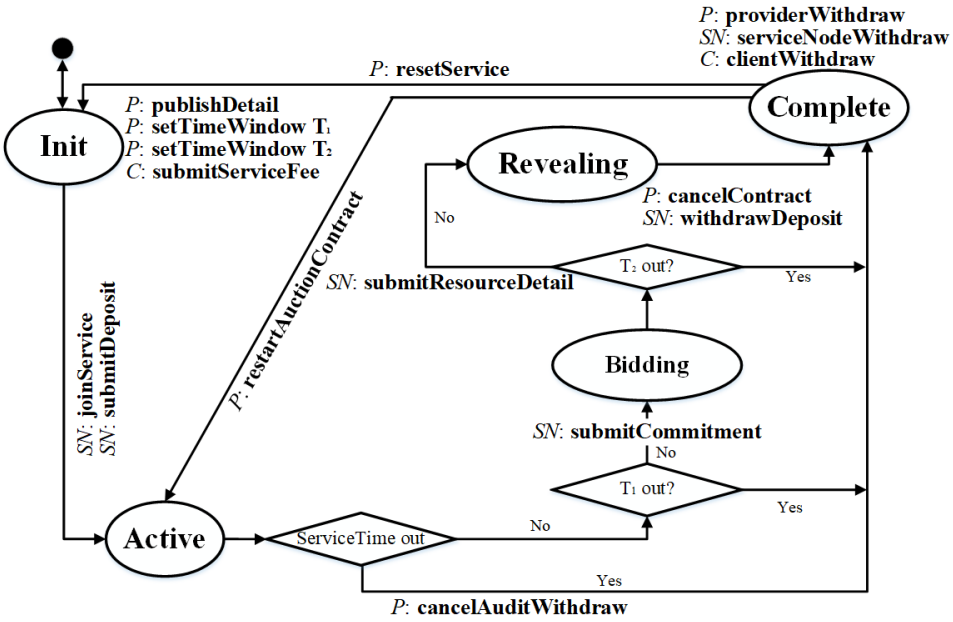


FIGURE 5 Audit task smart contract implementation.

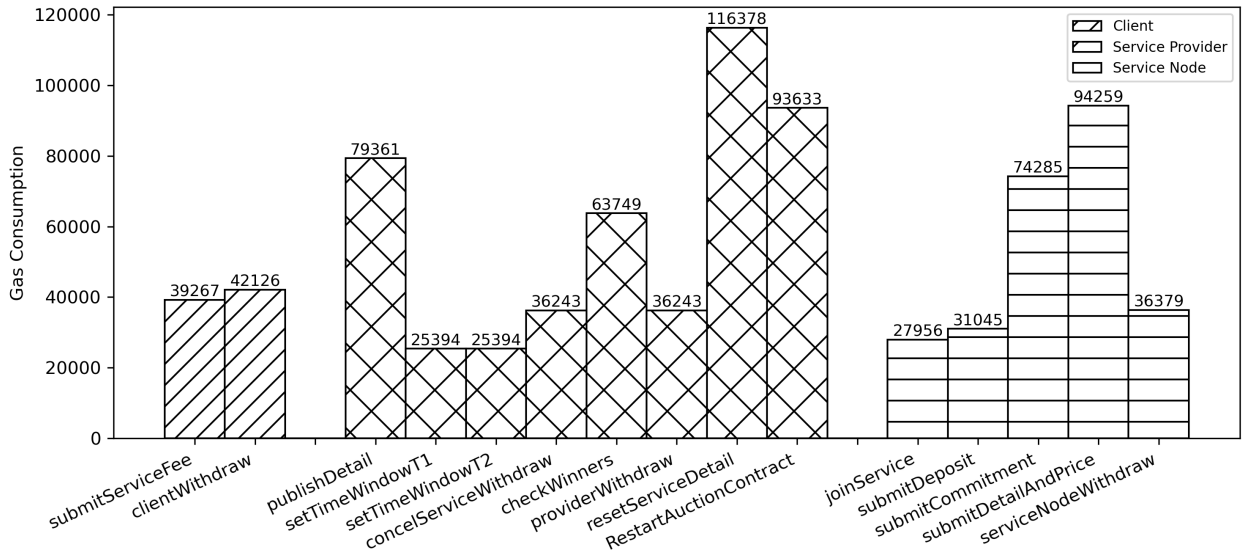


FIGURE 6 The Gas consumption of each interface in audit task smart contract.

In Ethereum, instead of running automatically, smart contracts can only be triggered and executed through interfaces. The transition of a smart contract from one state to another needs to be triggered by calling an interface function. Also, the implementation of interface functions in smart contracts requires a certain fee to be paid to Ethereum. For example, to switch from the "Bidding" state to the "Reveal" state, *SN* is required to submit the specific information of the quotation, which is to trigger

the state transition through the interface function "*SN*: submitResouceDetail". After the state transition is completed, the smart contract needs to pay a certain amount of Ether to Ethereum.

6.1.2 | Consumption of Each Interface in Auction Smart Contracts

We use Online Ethereum Studio[2] to develop Ethereum smart contracts. Online Ethereum Studio supports automatic deployment of simulated contracts with multi-user establishment, and users can interact with contracts through the Web client. Users can also test the usage of the interface in the contract, that is, the transaction fee and the cost of the Ethereum cryptocurrency "Ether". In order to simulate actual application scenarios, we test all interfaces of the auction model by deploying contracts on the Ethereum blockchain test network "Rinkeby". Rinkeby is a global blockchain test network for developers to debug smart contracts. It can simulate the secret currency "Ether" in the real Ethereum blockchain, which is convenient for us to debug the interface. We generate multiple accounts on "Rinkeby" to simulate different roles, and use the "Ether" retrieved from each simulated account to execute the interface and prepay different types of fees. The service information submitted by the client triggers the service party to establish a smart contract and deploy the contract to the blockchain. The service node chooses to join Crowd for bidding through the task details. Run all interfaces through different situations to realize functional testing and data collection of all interfaces.

Based on the economic game model of edges and clients, reasonable quotations for different resources are determined. The edges submit all quotations at once within a certain period through a two-stage submission strategy. The economic game model ensures that each edge makes quotations from the perspective of the greatest benefit; the two-stage submission strategy ensures fairness and rationality in the auction process and prevents the edge from arbitrarily changing prices or tampering with quotations through late submission. Therefore, we mainly study and analyze some performance information from the deployment and application of smart contracts. The complexity of the interface in the smart contract determines the performance of the operation, and the complexity of the interface determines the cost of state transition. In the Ethereum network, miners need to consume power to execute the procedures defined in the interface for verification and state conversion. Therefore, the more complex the interface, the higher the cost to pay. The "Gas" defined in Ethereum is used to measure execution costs. Transaction cost is the product of the consumption of "Gas" and the price per unit of "Gas". Therefore, the "Gas" consumption in the test network is similar to the consumption in the main network. Therefore, we recorded all the "Gas" consumption of each interface from the transaction records of the experiment to reflect the performance of our interface.

Figure 6 shows the amount of "Gas" in the main interface of the contract. The results show that the supplier's interface consumes more gas. Since the supplier needs to issue a smart contract and set some details in the contract, some of the interfaces run by the supplier also include the processing of the submitted data, so the "Gas" consumption is larger. Besides, the consumption of "Gas" mainly depends on the definition and implementation of the interface. More optimized and reasonable interface design can further reduce the consumption of "Gas".

6.2 | Cluster-based Block Distribution Algorithm Analysis

6.2.1 | Algorithm complexity analysis

The complexity of the cluster-based data distribution algorithm mainly consists of two parts, one is the clustering process, and the other is the process of selecting nodes from each cluster. Below we analyze the two processes and the overall complexity.

In the clustering process, the algorithm we choose is the KMeans clustering algorithm. The time complexity of KMeans is $O(l * n * k * m)$, and the space complexity is $O(n * m)$. Among them, m is the number of element fields, n is the amount of data, l is the number of iterations, and k is the number of clusters. At the same time, generally, l, k, m can be regarded as constants, so the time and space complexity can be simplified as $O(n)$, that is, the synthesis of the complexity of the KMeans algorithm is linear.

In the second process, the edges are selected from each cluster, mainly by performing a loop in each cluster. After clustering, the number of clusters is fixed, so it is equivalent to traversing all the data once. Therefore, the time complexity of the second process is also $O(n)$, which is also linear.

[2]https://studio.ethereum.org/

TABLE 2 Results of reliability measures.

Algorithm	Cluster number	Average distance	Shortest distance
Direct random selection	1	7.7985	1.0309
Random selection in each cluster	7	8.6452	3.2717
Our cluster based method	7	10.0075	4.4857

6.2.2 | Algorithm Comparison Analysis

In this section, we mainly compare the method of the random selection directly, random selection in each cluster after each clustering, and our cluster-based selection. We take the longitude and latitude as (120, 30) as the center and randomly generate 50 positions within a certain range. We set the block number as 7 and the cluster number is also 7. For random selection, the number of clusters can be regarded as only 1. Then, we respectively calculate the sum of the distances of the two positions and the minimum distance in each group as the measurement standard. The greater the distance sum and the greater the minimum distance, the better the file dispersion. The results are shown in Table 2 . For each item, we tested 20 rounds and calculated the average value. It can be seen from the table that our clustering-based algorithm takes the distance between the selected edge and the unselected cluster as the selection condition when selecting, and the selected result is better than the other two methods.

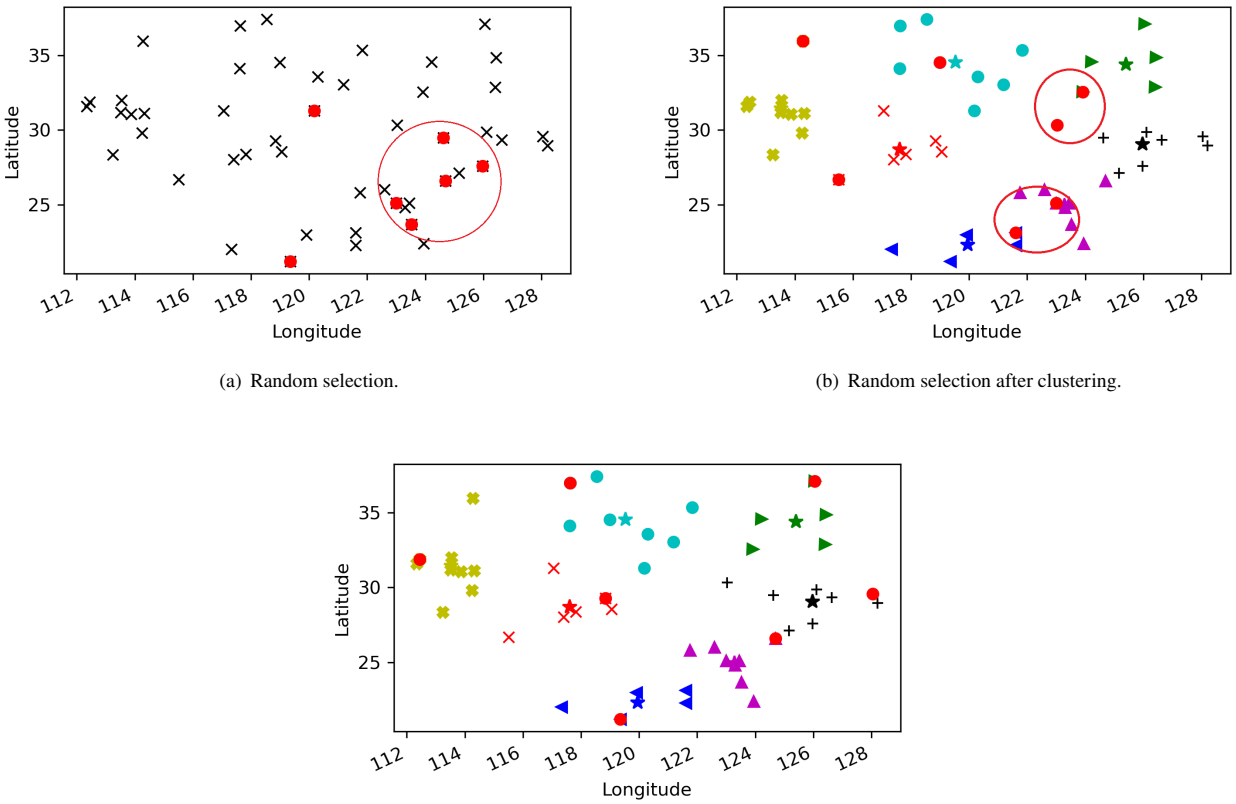
The distribution of the randomly generated data set is shown in Figure 3 . The distribution of red points is the distribution of selected edges when using different methods and the five-pointed stars with different colors represent the centroid of each cluster in Figure 7 . From Figure 7 , we can see the characteristics of edges distribution very intuitively. In Figure 7 (a), the selected edges are clustered in a certain geographic area. As we marked with a red circle in the figure, the attack on this area is likely to cause these edges to be dropped at the same time. For the random selection after clustering, edges selected in each cluster may result in the selected edges being located at the junction of two adjacent clusters, making an unwise choice, as shown in Figure 7 (b) where we use the red circles marking it out. In terms of location, two edges may be dropped by the same abnormal event, which reduces the reliability of file storage. In contrast, the edges selected using our cluster-based method in Figure 7 (c) can be well distributed in the region. The results in Figure 7 and the results in Table 2 can be mutually confirmed.

6.2.3 | Reliability Analysis

We assume that the different clusters have the same probabilities of being attacked due to external forces, denoted as p. We represent a File as F and the file F are consisted of n blocks, where m blocks in n blocks are redundant blocks, $n > m > 0$. In extreme cases, if all or n blocks are gathered in the same cluster. The probability of data loss is p. If all blocks are distributed in different clusters, the blocks number of being damaged are $n * p$. If more than m blocks are lost, the file F is lost and cannot be recovered. For a better explanation, we assume that a data composed of 4 original data blocks and 3 redundant blocks, data will be lost only when more than 3 blocks are lost. Besides, to simulate the distribution area of edges, we randomly generate a series of edge locations and describe their locations as shown in Figure 7 (b) and 7 (c). These generated position points represent the winning points in the auction process. The locations with different colors are clustered by KMeans. We can calculate the loss probability of the file F by the following formula:

$$\begin{aligned} P_{lost} &= C_n^{m+1} p^{m+1} (1-p)^{n-m-1} + C_n^{m+2} p^{m+2} (1-p)^{n-m-2} + \cdots + C_n^n p^n (1-p)^{n-n} \\ &= \Sigma_{i=m+1}^{n} C_n^i p^i (1-p)^{n-i} (m, n, i \in \mathbb{N}^+, 0 \le p \le 1). \end{aligned}$$

The result is shown in Table 3 . Obviously, $\Sigma_{i=m+1}^{n} C_n^i p^i (1-p)^{n=i}$ keeps decreasing with the increase of redundant data blocks and is much smaller than p. The probability of data loss is much smaller than that if the blocks are stored in a gathering distribution. Therefore, The decentralized distribution of blocks greatly improves the reliability of edge storage. Besides, we show in Figure 8 the file loss probability under different conditions when the number of original blocks is 15. We can intuitively observe that when the redundant data block is 1, the probability of file loss in dispersed storage is greater than the probability of gathered storage. This is because the file is stored in 16 locations, and any loss of 2 blocks can cause the loss of the file. With



(a) Random selection.

(b) Random selection after clustering.

(c) Our cluster-based selection method.

FIGURE 7 The distribution of selected edges using different methods.

the increase of redundant data blocks, the probability of file loss under decentralized storage continues to decrease until it is far less than that of gathered storage. This phenomenon is confirmed by the two situations (6,1) and (7,1) in the table 3 . Choosing an appropriate redundancy ratio can greatly improve the reliability of file storage.

6.2.4 | Time Consumption

To test the time consumption of the cluster-based selection algorithm, we conduct experiments on the Windows PC with 64-bit intel-core i5-8250U CPU at 1.8 GHz and 8 GB memory. Besides, we test each result 15 times and average the experimental results.

As shown in Figure 9 , we test the time consumption in different situations. First, we take the number of edges from 50 to 2000 and set the number of clusters to 50 and 100 respectively. We test time consumption to run each cluster-based clustering algorithm. We can get that as the number of edges increases, the time consumption also increases linearly as the results shown in Figure 9 (a). Besides, setting the number of edges to 2000 and 3000 respectively, we continued to test the time consumption of running the algorithm in the range of the number of clusters from 50 to 450. Similarly, for the same number of edges, the time consumption also increases as the number of clusters increases, and the results are shown in 9 (b).

Besides, in an actual system, the size of the data and the number of clusters can be specifically selected for the performance of the terminal device. In the case of too much data on the edges, some edges can also be selected in advance through some specific attributes required. For example, prefer to select the edge with fast access speed or large storage space.

TABLE 3 Probability of file loss under different aggregation situations.

$n-m$	n	m	**Gathered**	**Dispersed**	$n-m$	n	m	**Gathered**	**Dispersed**
3	4	1	0.1	0.0523	6	8	2	0.1	0.03809
3	5	2	0.1	0.00856	6	8	3	0.1	0.00833
4	5	1	0.1	0.08146	6	10	4	0.1	0.001635
4	6	2	0.1	0.01585	6	11	5	0.1	0.0002957
4	7	3	0.1	0.002728	7	8	1	0.1	0.186895
5	6	1	0.1	0.025692	7	8	2	0.1	0.07019
5	7	2	0.1	0.00502	7	10	3	0.1	0.012795
5	8	3	0.1	0.00089	7	11	4	0.1	0.00275
5	9	4	0.1	0.00089	7	12	5	0.1	0.0005412
6	7	1	0.1	0.14969	7	13	6	0.1	$9.9285 * 10^{-5}$

We set probabilities of each cluster being attacked, $p = 0.1$, n is the total blocks number, m is redundant blocks number. $n - m$ is the original blocks number.

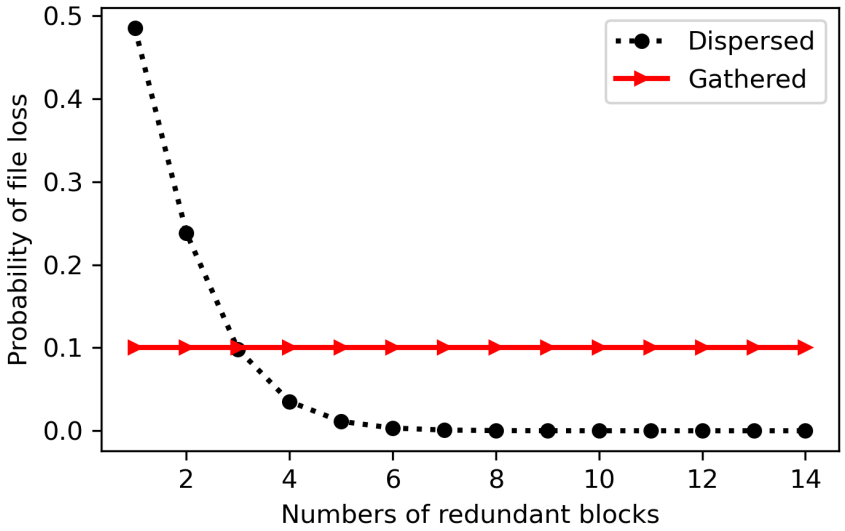
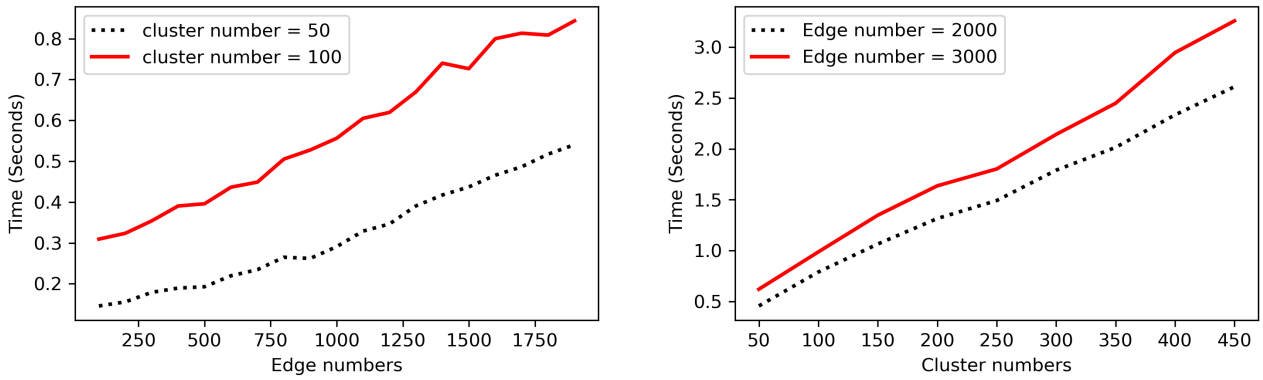


FIGURE 8 The Gas consumption of each interface in audit task smart contract.

7 | CONCLUSIONS

Through model construction and solution, mechanism design, implementation, and evaluation, we provide an edge storage solution named SmartStore. The SmartStore uses smart contracts to replace the traditional centralized data distribution platform. Besides, we propose using the game theory to conduct price analysis from an economic perspective and using a two-stage submission strategy to ensure the fairness of the SmartStore. After all edges and clients make reasonable prices, the edges that pass the auction are re-selected using improved clustering-based clustering methods to determine the final selected edges. SmartStore provides users with recommendations for storage solutions that take into account maximum revenue and storage reliability. Furthermore, some content can be further studied in the future. For example, the interfaces of the smart contract can be further improved to increase the operating speed and reduce "Gas" consumption. In this paper, the main idea is to provide solutions for edge storage that take into account the prices of heterogeneous resources and storage reliability and leave users with more independent customization options.



(a) Time consumption when edges'number increase.

(b) Time consumption when cluster's number increase.

FIGURE 9 Time consumption in different situations.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China(62072465), the National Key Research and Development Program of China(2018YFB0204301, 2020YFC2003400) and the NUDT Research Grants(No. ZK19-38).

References

1. Cisco C. Cisco Global Cloud Index: Forecast and Methodology, 2016–2021. *San Jose: Cisco.* 2018;.
2. Shi Weisong, Cao Jie, Zhang Quan, Li Youhuizi, Xu Lanyu. Edge computing: Vision and challenges. *IEEE internet of things journal.* 2016;3(5):637–646.
3. Xia Xiaoyu, Chen Feifei, He Qiang, Grundy John C, Abdelrazek Mohamed, Jin Hai. Cost-effective app data distribution in edge computing. *IEEE Transactions on Parallel and Distributed Systems.* 2020;32(1):31–44.
4. Qiao Fuli, Wu Jun, Li Jianhua, Bashir Ali Kashif, Mumtaz Shahid, Tariq Usman. Trustworthy edge storage orchestration in intelligent transportation systems using reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems.* 2020;:1-14. https://doi.org/10.1109/TITS.2020.3003211.
5. Lin Hsiao-Ying, Tzeng Wen-Guey. A secure decentralized erasure code for distributed networked storage. *IEEE transactions on Parallel and Distributed Systems.* 2010;21(11):1586–1594.
6. Liang Lixin, He Huan, Zhao Jian, Liu Chengjian, Luo Qiuming, Chu Xiaowen. An Erasure-Coded Storage System for Edge Computing. *IEEE Access.* 2020;8:96271–96283.
7. Mayer R., Gupta H., Saurez E., Ramachandran U.. FogStore: Toward a distributed data store for Fog computing. In: 2017 IEEE Fog World Congress (FWC); Oct.30 - Nov.1, 2017; Santa Clara, CA, USA. https://doi.org/10.1109/FWC.2017.8368524.
8. Psaras Ioannis, Ascigil Onur, Rene Sergi, Pavlou George, Afanasyev Alex, Zhang Lixia. Mobile Data Repositories at the Edge. In: USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18); July, 2018; Boston, MA. https://www.usenix.org/conference/hotedge18/presentation/psaras.
9. Gedeon Julien, Himmelmann Nicolás, Felka Patrick, Herrlich Fabian, Stein . vStore: A context-aware framework for mobile micro-storage at the edge. *International Conference on Mobile Computing, Applications, and Services.* 2018;240:165–182.
10. Chen Chien-An, Won Myounggyu, Stoleru Radu, Xie Geoffrey G. Energy-efficient fault-tolerant data storage and processing in mobile cloud. *IEEE Transactions on cloud computing.* 2014;3(1):28–41.

11. Yu Jiaping, Chen Haiwen, Wu Kui, Zhou Tongqing, Cai Zhiping, Liu Fang. Centipede: Leveraging the Distributed Camera Crowd for Cooperative Video Data Storage. *IEEE Internet of Things Journal.* 2021;preprint:1–1. https://doi.org/10.1109/JIOT.2021.3074823.

12. He Shuqing, Cheng Bo, Wang Haifeng, Xiao Xuelian, Cao Yunpeng, Chen Junliang. Data security storage model for fog computing in large-scale IoT application. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS); April 15-19, 2018; Honolulu, HI, USA. https://doi.org/10.1109/INFCOMW.2018.8406927.

13. Linaje Marino, Berrocal Javier, Galan-Benitez Alfonso. Mist and edge storage: Fair storage distribution in sensor networks. *IEEE Access.* 2019;7:123860–123876.

14. Moysiadis Vasileios, Sarigiannidis Panagiotis, Moscholios Ioannis. Towards distributed data management in fog computing. *Wireless Communications and Mobile Computing.* 2018;2018.

15. Zhang Qikun, Li Yongjiao, Wang Ruifang, Liu Lu, Tan Yu-an, Hu Jingjing. Data security sharing model based on privacy protection for blockchain-enabled industrial Internet of Things. *International Journal of Intelligent Systems.* 2021;36(1):94–111.

16. Borjigin Wuyunzhaola, Ota Kaoru, Dong Mianxiong. In broker we trust: A double-auction approach for resource allocation in NFV markets. *IEEE Transactions on Network and Service Management.* 2018;15(4):1322–1333.

17. Jin A-Long, Song Wei, Wang Ping, Niyato Dusit, Ju Peijian. Auction mechanisms toward efficient resource sharing for cloudlets in mobile cloud computing. *IEEE Transactions on Services Computing.* 2015;9(6):895–909.

18. Kiani Abbas, Ansari Nirwan. Toward hierarchical mobile edge computing: An auction-based profit maximization approach. *IEEE Internet of Things Journal.* 2017;4(6):2082–2091.

19. Sun Wen, Liu Jiajia, Yue Yanlin, Zhang Haibin. Double auction-based resource allocation for mobile edge computing in industrial internet of things. *IEEE Transactions on Industrial Informatics.* 2018;14(10):4692–4701.

20. Bahreini Tayebeh, Badri Hossein, Grosu Daniel. An envy-free auction mechanism for resource allocation in edge computing systems. In: 2018 IEEE/ACM Symposium on Edge Computing (SEC); Oct. 25-27, 2018; Seattle, WA, USA. https://doi.org/10.1109/SEC.2018.00030.

21. Peng Xiting, Ota Kaoru, Dong Mianxiong. Multiattribute-based double auction toward resource allocation in vehicular fog computing. *IEEE Internet of Things Journal.* 2020;7(4):3094–3103.

22. Myerson Roger B. *Game theory*. Harvard university press; 2013.

23. Li Ming-Chu, Xu Lei, Sun Wei-Feng, Lu Kun, Guo Cheng. Grid resource allocation model based on incomplete information game. *Ruanjian Xuebao/Journal of Software.* 2012;23(2):428–438.

24. Pillai Parvathy S, Rao Shrisha. Resource allocation in cloud computing using the uncertainty principle of game theory. *IEEE Systems Journal.* 2014;10(2):637–648.

25. Guo Jian, Liu Fangming, Zeng Dan, Lui John CS, Jin Hai. A cooperative game based allocation for sharing data center networks. In: 2013 Proceedings IEEE INFOCOM; April 14-19, 2013; Turin, Italy. https://doi.org/10.1109/INFCOM.2013.6567016.

26. Ho Tai Manh, Tran Nguyen H, Kazmi SM Ahsan, Hong Choong Seon. Dynamic pricing for resource allocation in wireless network virtualization: A Stackelberg game approach. In: 2017 International Conference on Information Networking (ICOIN); Jan. 11-12, 2017; Da Nang, Vietnam. https://doi.org/10.1109/ICOIN.2017.7899529.

27. Li Qihui, Huang Chuanhe, Bao Haizhou, Fu Bin, Jia Xiaohua. A game-based combinatorial double auction model for cloud resource allocation. In: 2019 28th International Conference on Computer Communication and Networks (ICCCN); July 29-Aug. 1, 2019; Valencia, Spain. https://doi.org/10.1109/ICCCN.2019.8846922.

28. KN Shyleshchandra Gudihatti, Roopa MS, Tanuja R, Manjula SH, Venugopal KR. Energy aware resource allocation and complexity reduction approach for cognitive radio networks using game theory. *Physical Communication.* 2020;42:101152.

29. Noor Mohammed Vali Mohamad, Sreenivasan Prithvi Mothi, Ravishankar Tharrun, Hariharan Subramaniyam, Lakshmanan Muthukaruppan. Energy-efficient resource allocation for device-to-device communication through noncooperative game theory. *International Journal of Communication Systems.* 2020;33(6):e4279.

30. Wu Hongjia, Zhang Jiao, Cai Zhiping, et al. Resolving Multi-task Competition for Constrained Resources in Dispersed Computing: A Bilateral Matching Game. *IEEE Internet of Things Journal.* 2021;preprint:1–1. https://doi.org/10.1109/JIOT.2021.3075673.

31. Xiao Yang, Zhang Ning, Li Jin, Lou Wenjing, Hou Y Thomas. PrivacyGuard: Enforcing Private Data Usage Control with Blockchain and Attested Off-Chain Contract Execution. In: European Symposium on Research in Computer Security; September 13, 2020. https://doi.org/10.1007/978-3-030-59013-0_30.

32. Fehér D. J., Sándor B.. Log File Authentication and Storage on Blockchain Network. In: 2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY); September 13-15, 2018; Subotica, Serbia. https://doi.org/10.1109/SISY.2018.8524848.

33. Chatzopoulos Dimitris, Ahmadi Mahdieh, Kosta Sokol, Hui Pan. Flopcoin: A cryptocurrency for computation offloading. *IEEE transactions on Mobile Computing.* 2018;17(5):1062–1075.

34. Kopp Henning, Mödinger David, Hauck Franz, Kargl Frank, Bösch Christoph. Design of a privacy-preserving decentralized file storage with financial incentives. In: 2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW); April 26-27, 2017; Paris, France. https://doi.org/10.1109/EuroSPW.2017.45.

35. Backman Jere, Yrjölä Seppo, Valtanen Kristiina, Mämmelä Olli. Blockchain network slice broker in 5G: Slice leasing in factory of the future use case. In: 2017 Internet of Things Business Models, Users, and Networks; Nov. 23-24, 2017; Copenhagen, Denmark. https://doi.org/10.1109/CTTE.2017.8260929.

36. Kang Jiawen, Yu Rong, Huang Xumin, Maharjan Sabita, Zhang Yan, Hossain Ekram. Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains. *IEEE Transactions on Industrial Informatics.* 2017;13(6):3154–3164.

37. Harsanyi John C, Selten Reinhard. A generalized Nash solution for two-person bargaining games with incomplete information. *Management science.* 1972;18(5-part-2):80–106.

38. Bertoni Guido, Daemen Joan, Peeters Michaël, Van Assche Gilles. Keccak. In: Annual international conference on the theory and applications of cryptographic techniques; 2013; Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-38348-9_19.

Algorithm 1 *Joint Attribute and Reliability Selection Algorithm*

Input:

Blocks number k;
Resource number t;
m edges: $E_m = \{e_1, ..., e_m\}$;
Location set of E_m: $\{L_1, L_2, ..., L_m\}$, L_i=$\{latitude_i, longitude_i\}$, $1 \leq i \leq m$;
Preferred resource of edges: a;

Output:

Set of selected edges: $selectedSet$;

Random select k items from S_m as starting centroid $C_k = \{c_1, ..., c_k\}$;
C_k corresponds to k clusters: $Cluster_k = \{cluster_1, ..., cluster_k\}$;
while The cluster state has changed and the maximum iterations number has not been reached **do**
    for each item e_i in $E_m = \{e_1, ..., e_m\}$ **do**
        Distance[] $DisArr$;
        Distance[][][] $DisArrs$;
        for each item c_j in C_k **do**
            Distance $d_j = Dis(e_i, c_j) = Dis(L_i, L_j)$;
            $DisArr \leftarrow d_j$;
            $DisArrs[i][j] \leftarrow d_j$;
        end for
        Shortest distance $d_s = minDis(DisArr)$;
        Corresponding cluster $cluster_{d_e}$;
        $cluster_{d_e} \leftarrow e_i$;
    end for
    for each $cluster_i$ in $Cluster_k$ **do**
        Update centroid: $c_i \leftarrow getCentroid(cluster_i)$;
    end for
end while
Selected edges: $selectedSet = \emptyset$;
The Clusters of selected edges set P;
The Clusters of remaining to be select: $\hat{P}$;
# Select some preferred edges according to the attribute.
for each $cluster_i$ in $Cluster_k$ **do**
    $P, \hat{P}, selectedSet = getOptimal(cluster_i, a)$;
end for
# Select edges from remaining cluseters $\hat{P}$.
for each $cluster_j$ in $\hat{P}$ **do**
    Distance[] $DisArrTemp = \emptyset$;
    # Compute the distance between each edge and selectSet and get the shortest one.
    for each e_k in $cluster_j$ **do**
        $DisArrTemp \leftarrow minDis(DisArrs, selectedSet)$;
    end for
    $selectedSet \leftarrow maxDis(DisarrTemp)$;
end for
Return $selectedSet$

How to cite this article: H. Chen, J. Yu, H. Zhou, T. Zhou, F. Liu and Z. Cai (2021), SmartStore: A Blockchain and Clustering Based Intelligent Edge Storage System with Fairness and Resilience, *Internation Journal of Intelligent Systems*, *2021;00:1–22*.