# TaYi-Future-oriented value blockchain network

Author: YeMao Xiao

## Abstract

With the development of blockchain technology and the rise of non-homogeneous tokens (NFTs), this study proposes an innovative blockchain architecture aiming to improve the data processing flexibility of blockchains by using NFTs as core data storage and management elements. In this architecture, NFT not only acts as the chain state basic unit, but also plays the role of a node in the graph structure maintained by the blockchain, which optimizes the efficiency of data organization and retrieval. In addition, smart contracts, account information, and other blockchain data are abstracted into NFT structures to enhance the security and transparency of the system. This study adopts a hybrid methodology for design and implementation, which first elaborates the NFT-based blockchain design principles and data structures, and then implements smart contracts through real-world cases to verify their usability. The experimental results show that the design not only improves the credibility of data retrieval results, but also enhances the scalability of the blockchain system. This study provides new perspectives and solutions for blockchain technology in dealing with complex data structures and optimizing systematic data retrieval, which is of great significance for the future application and research of dedicated blockchain and NFT.

# Introductory

Blockchain technology is a distributed ledger formed by using cryptography to append consensus-confirmed blocks in a sequential order released by Satoshi Nakamoto in 2009 [1].

With the development of blockchain technology, thanks to the decentralized and tamper-proof characteristics of blockchain [2], blockchain has already played a great role in the fields of finance, supply chain, and education [3-7].

With the continuous development of blockchain technology, especially the technological development of smart contracts based on Ether, people are not only limited to creating homogeneous tokens, but also turn their attention to more unique non-homogeneous tokens, and smart contracts allow developers to create unique and irreplaceable digital assets based on smart contract technology zones, and NFTs are born [8].

Used primarily in the digital art and collectibles space, NFT can provide a trusted way of verifying the authenticity and ownership of digital artworks, which was previously difficult to achieve in the traditional digital environment. In addition, a key feature of NFT is its decentralization, which is inherited to the decentralized nature of blockchain and ensures that NFT's ownership record and verification operations will not rely on any centralized organization, but rather be implemented through the underlying blockchain network, which greatly improves the credibility of NFT's data records.

With the technological development of Ethernet smart contracts and the expansion of the blockchain, more and more data will be allowed to be stored on the blockchain, and also because of the uniqueness and non-interchangeability brought about by the ownership property of the

NFT structure, the NFT structure has been better applied and adapted in various fields.

By allowing NFTs to be used as collateral for instant loans, NFThis borrows the decentralization and value of NFTs, reduces the intermediate operations between the borrower and the lender, and relies on the decentralized ledger nature of the blockchain to guarantee the security and transparency of the loan process. [9]

An NFT-based mechanism for the classification and disposal of healthcare waste is proposed to set up incentives and penalties for organizations and individuals, to provide recommendations for the first stage of healthcare waste disposal (i.e., garbage sorting), and to raise individual and collective awareness of garbage sorting in healthcare environments. [10]

A medical data sharing system based on the NFT combinatorial model was constructed to improve the security of patients' personal information and to facilitate the sharing of data documents in the healthcare environment. [11]

Leveraging NFT to tokenize real-world supporting documents such as driver's licenses and university certificates to prevent fraud and increase transparency in the authentication space. Increased trust, transparency and security demonstrates the great role of NFT in the field of document authentication. [12]

However, most of the current applications for NFT are based on the special structure implemented by the smart contract provided by the blockchain, where the structure of the NFT nodes that can be stored in the smart contract is simpler, and most of the data still relies on other storage structures, such as decentralized data storage applications like IPFS.

Studies have shown that these types of methods can have a certain

probability of failure during storage, leading to a loss of value of the NFT.

Second, in the smart contract, in order to regulate the consumption of the gas fee, the blockchain sets the maximum gas fee now, but some core NFT operations, such as judging the NFT status and obtaining the list of all NFT data of a specified user, often require cyclic operations, which are not recommended by the smart contract. In order to carry out these operations, workers generally adopt the scheme of off-chain centralized synchronization of on-chain data for querying, however, this type of scheme has certain drawbacks.

The solution is technically implemented in the service provider to build a centralized database, the user from the centralized database to obtain the final required data, the threat of tampering with the service provider's data exists, often resulting in the user's final access to the data credibility is greatly reduced.

The service provider data nodes in this scheme are unable to synchronize all the NFT structural data on the chain and data ownership change operations in a timely manner.

The on-chain processing is not flexible enough to perform more complex operations, such as constructing a system of relationships between NFTs.

In order to solve the above drawbacks, this paper proposes a blockchain based on ERC721 standard [13] with NFT data structure and NFT relationship system.

# Related work

From a technical perspective, a cryptocurrency ledger such as Bitcoin can be viewed as a state-transition system that has a "state" consisting of the ownership state of all existing Bitcoins and a "state transition function" that takes the state and transactions as input and outputs a new state as a result. The state transition function takes state and transactions as input and outputs a new state as a result.

Bitcoin, the first cryptocurrency ledger, also has a relatively simple state transition mechanism. He maintains an existing bitcoin ownership state, and when a transfer is sent, the existing state and the transaction are fed into a state transition function, which performs a state transition and uses the output as the new state.

Ether, the emerging cryptocurrency ledger, categorizes state into accounts, which are divided into external accounts and contract accounts. External accounts are controlled by private keys, and holders can send messages from external accounts by creating and signing transactions. Contract accounts store a fixed, tamper-proof piece of code that activates internal storage permissions whenever a transaction is received by the contract account, and sends other messages and creates contracts.

Bitcoin maintains a currency ownership state and relies on state changes performed by state functions to operate. Ether maintains two state identities on top of the currency ownership state, an external account similar to Bitcoin's, and a contractual account containing code that is not considered to be under control. However, the relationship and expression of value between the two accounts is ignored, and the internal storage data written is decentralized and not easily linked

and utilized by other accounts on the chain that would bring more flexibility and control.

In specific examples, such as the on-chain storage of value data, developers want to have more efficient and convenient data manipulation to assist on-chain operations, and all data should not be closed, but should be open and shared and self-governing in the same way as the values of the blockchain.

In a data ownership structure, a monetary ledger maintained for state changes brought about by homogenized tokens would essentially contain only transfer relationships between individual accounts, and the code storage and activation strategies brought about by contract-based accounts would contain only the account's own state changes and those of the homogenized tokens. These state change scenarios would not be applicable to constructing operations on the chain with more ownership attributes, nor would they be applicable to the maintenance of more complex state change data.

Thanks to the uniqueness and ownership attributes of non-homogenized tokens, richer state variations will be maintained, and the ownership attribute of NFT can be used to construct currency ownership states, and the value relationship states based on other relationship attributes of NFT can be constructed, and at the same time, all the attributes of the blockchain of homogenized token states can be inherited.

# NFT structure with more value and relationship properties

## ERC721

ERC-721 is an important Ethereum Improvement Proposal (EIP) that

defines a Non-Fungible Token (NFT) standard. This standard was proposed in 2018 by William Entriken, Dieter Shirley, Jacob Evans, and Nastassia Sachs, and is officially documented as EIP-721.

The core of the ERC-721 standard is the introduction of a type of token on the blockchain that represents a unique asset. These tokens are referred to as non-homogenized because each token is unique and has different properties or values from other tokens. This is in contrast to traditional cryptocurrencies or homogenized tokens (such as those under the ERC-20 standard), where each unit is fungible and equivalent.

The ERC-721 standard provides a mechanism for digitizing, verifying, and transacting a wide range of unique assets such as digital collectibles, works of art, virtual properties, in-game items, and more. With this standard, ownership and transfer of assets can be securely and transparently recorded and verified on the Ethernet blockchain.

On a technical level, ERC-721 defines a number of interfaces, including functions for tracking token owners and functions for transferring tokens. These interfaces ensure the standardization of NFTs, allowing different applications, marketplaces and services to interoperate and support ERC-721 tokens.

## ERC1155

ERC-1155 is an advanced Ethereum Improvement Proposal (EIP) that aims to provide a more flexible and efficient token standard for many different asset types. The standard was co-proposed by Witek Radomski, Andrew Cooke, Philippe Castonguay, James Therien, Eric Binet, and Ronan Sandford, and was officially documented as EIP-1155 in 2018.

In contrast to the ERC-721 standard (non-homogenized tokens) and the ERC-20 standard (homogenized tokens), ERC-1155 introduces a unique

approach to handling multiple different types of tokens, whether homogenized or non-homogenized. This is one of the main innovations of ERC-1155, enabling it to support multiple types of tokens simultaneously in a single smart contract.

The core advantage of the ERC-1155 standard is its high degree of flexibility and efficiency. For example, in games, developers can use a single ERC-1155 contract to represent various assets, such as currencies, items, characters, etc., regardless of whether they are unique (non-homogeneous) or interchangeable (homogeneous). This significantly reduces the number of smart contracts that must be deployed, lowers transaction costs, and simplifies the transaction process.

In addition, ERC-1155 provides a more efficient transaction method. In ERC-721, each token transfer requires a separate transaction. In ERC-1155, on the other hand, it is possible to send and receive many different tokens simultaneously in a single transaction, which is particularly useful in scenarios where a large number of assets need to be processed in bulk.

## EIP6551

EIP-6551 is a proposal to assign Ether accounts to all non-homogenized tokens (NFTs). This system allows NFTs to own assets and interact with applications without changing existing smart contracts or infrastructure. The goal of the proposal is to give every NFT the same rights as an Ether user, including the right to self-custody of assets, to perform arbitrary operations, to control multiple separate accounts, and to use accounts on multiple chains. The proposal defines a single-case registry for managing these token-bound accounts and their interactions. This approach enables complex real-world assets to be represented as NFTs through a common schema that mirrors Ethernet's

existing ownership model.

| characterization | ERC721 | ERC1155 | EIP6551 |
|---|---|---|---|
| safety | surety | surety | surety |
| affiliation | 1:1 | 1:n | 1:n or 1:1 |
| scalability | non-scalable | scalable | scalable |
| Direct traceability up the chain | non-traceable | non-traceable | non-traceable |

ERC1155 provides a scheme for bulk digital asset management, however, in the management of actual non-homogenized tokens, the author believes that the uniqueness of the data needs to be immutable, and the existing scheme will lead to a reduction in the value of the data.ERC721 proposes a scheme for the management of a single digital asset, which well corresponds to the general management mode of real assets, and the data structure of this type can be better retrieval and uniqueness confirmation.EIP6551 provides a scheme combining NFT and traditional Ethernet accounts, which guarantees the uniqueness of data while providing a broader operation space for NFT structure.

# Methodology

Based on the above theory, ERC721 is combined with EIP6551 to provide an NFT structure that is easy to determine uniqueness and has a wide range of operating privileges.

$$NFT = (ID, Metadata, Owner, NftLabel)$$

ID: This is a unique identifier that uniquely identifies this particular NFT among all NFTs, similar to an account address

Metadata: this is a collection containing NFT descriptive information such as author, title, date of creation, numerical representation of the artwork, etc.

Owner: This represents the current owner of the NFT and is an account address that is recorded in the data and establishes an authorization relationship.

NftLabel: identifies the type of account, which can be external, contract, or data.

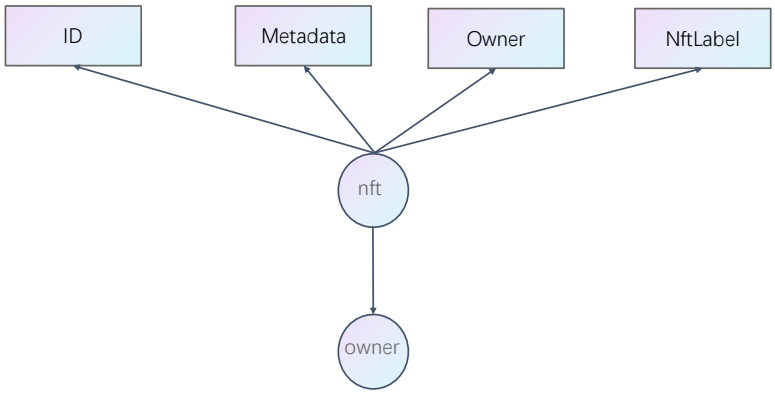


Chart 1 A more valuable NFT structure

## Storage abstraction

Ether's internal storage data is decentralized and not easily shared and utilized by other accounts, and can be specified again as account internal data and external data. Internal data is a storage structure that is consistent with the Ethernet contract account, when the contract account receives a transaction, it will give access to the account's internal storage to perform an operation, which is limited to the contract's internal data and cannot directly manipulate other contract data. External data is a type of data that distinguishes it from internal data, which will be a new type of account, i.e. data account. When the contract account code is activated, it will simultaneously activate the linking privileges of the external authorized data account, which can be operated. The data account will be stored in NFT format, and will be stored at the same level as the external and contract accounts, which are also NFT structured, and maintain their linkage map relationships.

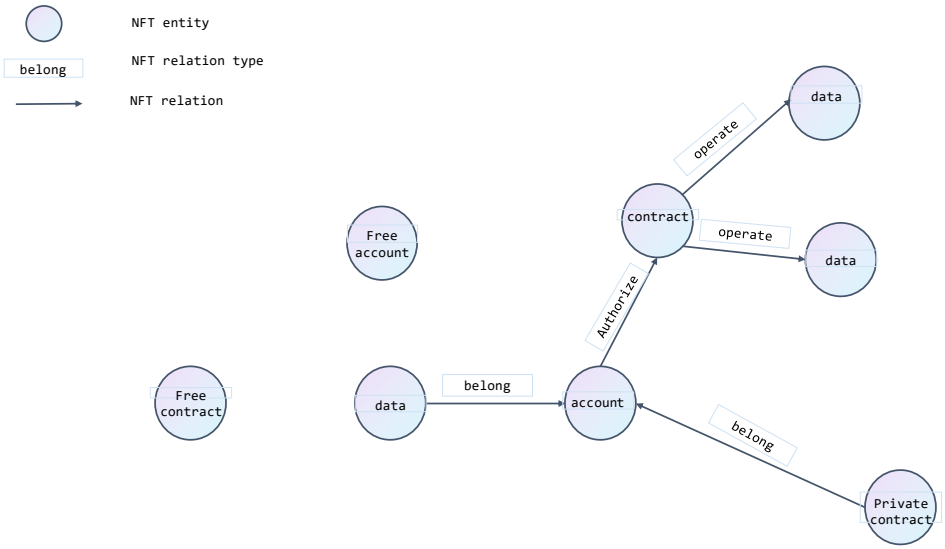


Chart 2 Storage abstraction principle

## Graphical abstraction of relations

In order to provide a more flexible NFT structure, all NFT data can be abstracted as graph structure data nodes, and certain connections between nodes can be abstracted as relationship representations, on the basis of which a data graph structure of all data and relationships can be maintained.

Taking the social relationship system as an example, there are existing external account A, external account B and smart contract C and smart contract D, in which A and B are social relationships, while C and D are social smart contracts. When A links with smart contract C, it needs to access A's social relationship, then it will determine whether C is authorized to link by A. If it is authorized, it will determine the data range that C is authorized to link. For example, C is authorized to access the permission of such data as 'social relationship' of A, then C will link the edge node of A's social relationship. When accessing the edge node, it will consider whether the B account authorizes C to access it, and if C is authorized to access it, then B will be linked by C and continue the operation.

Such strict state management will effectively protect data privacy while providing more flexible operations.

## Account system

In this system, accounts contain three categories: external accounts, contract accounts, and data accounts, and the management of the life cycle of the accounts will be carried out by the caster and linker.

The caster will complexly cast new NFT data such as new external accounts, new contract accounts, new data accounts. New nodes will be created in the data graph structure and casting will be done according

to predefined types to ensure the uniqueness and reliability of the casted accounts.

The linker will be responsible for the management of inter-account relationships, such as relationship creation, data connections, etc., to ensure the security and flexibility of accounts.

```
// Function to link an account to a smart contract
def link_account_to_contract(graph, account_id, contract_id).
    if contract_id in graph.nodes[account_id].relations::
        // The contract is already directly related to the account
        return True
    else.
        // Check if the contract is authorized to link to the account
        if check_authorization(graph.nodes[account_id],
graph.nodes[contract_id]).
            // If authorized, link the contract to the account's
social relations
            for related_id in graph.nodes[account_id].relations:
                if check_authorization(graph.nodes[related_id],
graph.nodes[contract_id]).
                    // Link the contract to the related account and
perform operations
                    graph.add_relationship(contract_id, related_id)
                    perform_operations(graph.nodes[contract_id],
graph.nodes[related_id])
        else.
            // If not authorized, deny the link
            return False

// Helper functions to check authorization and perform operations
```

```
def check_authorization(node1, node2).
    // Implement the logic to check if node2 is authorized to link to node1.
    // This can include checking if node2 has the permission to access node1's social relations.
    pass

def perform_operations(node1, node2).
    // Implement the logic for the operations to be performed once the link is established.
    pass

// Example usage
// Initialize a new Graph
nft_graph = Graph()

// Create Nodes for accounts and smart contracts
account_A = Node("A", "account")
account_B = Node("B", "account")
contract_C = Node("C", "contract")
contract_D = Node("D", "contract")

// Add Nodes to the Graph
nft_graph.add_node(account_A)
nft_graph.add_node(account_B)
nft_graph.add_node(contract_C)
nft_graph.add_node(contract_D)
```

```
// Define relationships between accounts
nft_graph.add_relationship("A", "B")

// Attempt to link account A to contract C
link_account_to_contract(nft_graph, "A", "C")
```

## Smart contract (finance)

Smart contracts are a type of automated protocol used in blockchain technology. They are programs stored on the blockchain that can automatically execute and enforce the terms of a contract when predefined conditions are met. This means that when a specific condition defined in the contract is triggered, the smart contract automatically performs the relevant action, such as transferring funds, recording data, or initiating other smart contracts.

The advantage of smart contracts is that they provide a secure, transparent and third-party intermediary-free way to process transactions and agreements. They are useful in a variety of applications, including financial services, supply chain management, automated governance systems and digital identity verification. Smart contracts help increase efficiency and reduce transaction costs by reducing disputes and increasing processing speed.

In this system, smart contracts are designed to be used in the management of states that will allow external accounts or contract accounts to access authorized data accounts. Such as adding new data accounts, authorizing data accounts, constructing account relationships, etc.

Based on the characteristics of the system, controlling the ownership will become incredibly important, in which smart contract accounts will be divided into public and private accounts, private

accounts will only receive transactions sent by the owner, while public accounts are the same as Ether, and can receive all transactions, the author believes that the ownership relationship of the smart contract will make the smart contract more valuable, which will be a bold attempt.

## state transition system

In the ownership state transition system based on homogenized tokens, the change of ownership state of the homogenized tokens will be input into the state transition function as a parameter, and the output of the state transition function will be used as the new state.

In the account state based transformation system, the state change of the account is input as a parameter to the state transformation function to take the output account state as the new state.

Inherit a state transition system for homogenized tokens, which will involve a state transition system for homogenized tokens, and will maintain the ownership state of non-homogenized tokens and the state of the chart of accounts data structure

This system state transition system is similar to the account state based transition system.

The state transition function $StatusChange(Status, TX) -> Status'$ can be defined as follows:

Checks that the operation is in the correct format, that the transaction signature is authentic, and that the Nonce is valid, and if the check fails, refuses to execute the transaction and returns an error.

Determine the account situation, if the recipient is a data account, it is necessary to determine the authorization situation of the data account, whether it can be accessed or not. This step mainly determines the data account authorization issue and determines the required checking cost according to the number of new accounts involved,

i.e. $COUNT * GASPRICE$ . If the cost is insufficient, it returns an error and pays the miner's checking cost.

Calculate the required gas fee by $STARTGAS * GASPRICE$ Calculating the transaction fee and determining the sender address from the signature. Subtracts the fee from the sender's account balance and increases the sender's nonce value. If the account balance is insufficient, an error is returned.

Based on the size of the data in the transaction to make a judgment on the fuel cost to be deducted, if the cost is insufficient, the execution of the transaction will be rejected and an error will be returned.

If the pre-check is ready, start executing the transaction, and if the sending address is a contract account, execute the contract account code until the fuel charge runs out or the code execution ends.

If the required cost exceeds the cost of executing the code, roll back all state except the executed code and send the code execution cost to the miner.

Finally, the miner's cost of executing the transaction is deducted to determine if there is a balance, and if so, the balance is returned to the sender.

```
function StatusChange(CurrentStatus, Transaction) -> NewStatus.
    // Check if the transaction format is correct
    if not IsValidFormat(Transaction).
        return Error("Transaction format is incorrect")

    // Verify the authenticity of the transaction signature
    if not IsSignatureAuthentic(Transaction).
        return Error("Transaction signature is not authentic")
```

```
    // Check if the nonce is valid
    if not IsNonceValid(Transaction).
        return Error("Nonce is invalid")

    // If the receiver is a data account, verify access authorization
    if IsDataAccount(Transaction.receiver).
        // Determine the cost of checks based on the number of new
accounts involved
        checkCost = COUNT(Transaction.newParticipants) * GASPRICE
        if not HasSufficientFunds(Transaction.sender, checkCost)::
            // Pay the miner the check fee if funds are insufficient
            PayMinerCheckFee(Transaction.sender, checkCost)
            return Error("Insufficient funds for data account
authorization check")

    // Calculate the required gas cost for the transaction
    gasCost = STARTGAS(Transaction) * GASPRICE
    // Deduct the transaction fee from the sender's account balance
    if not DeductFunds(Transaction.sender, gasCost)::
        return Error("Insufficient funds for gas")

    // Deduct fuel costs based on the size of data in the transaction
    fuelCost = CalculateFuelCost(Transaction.dataSize)
    if not HasSufficientFunds(Transaction.sender, fuelCost)::
        return Error("Insufficient funds for fuel cost")

    // Execute the transaction if all pre-checks are successful.
```

```
    if IsContractAccount(Transaction.sender).
        executionResult = ExecuteContractCode(Transaction.sender, Transaction.data)
        // If execution cost exceeds the provided gas, rollback and pay the miner
        if executionResult.cost > Transaction.providedGas:
            RollbackStateChanges()
            PayMinerExecutionFee(Transaction.sender, executionResult.cost)
            return Error("Gas exceeded during contract execution")

    // Deduct the miner's fee for executing the transaction
    DeductMinerFee(Transaction.sender, executionResult.minerFee)

    // Check for any remaining balance and return it to the sender
    remainingBalance = GetRemainingBalance(Transaction.sender)
    if remainingBalance > 0.
        RefundSender(Transaction.sender, remainingBalance)

    // Return the new status after successful execution
    return NewStatus
```

## Status Change Table

In the directed graph maintained by the blockchain, the relevant operations can perform the following state transitions, using the adjacency matrix.

Let us maintain a directed graph G and its adjacency matrix A, which has dimension i*j,i=j, indicating that there are i nodes in the graph.

$$G = (V, E)$$

V is a set representing all vertices (nodes) in the graph.

E is a set representing all directed edges in the graph

Among them.

$$E = (\mathrm{V}_i, \mathrm{V}_j, \mathrm{L}_{ij}), \ldots, (\mathrm{V}_x, \mathrm{V}_y, \mathrm{L}_{xy})$$

$\mathrm{V}_i, \mathrm{V}_j, \mathrm{V}_\mathrm{x}, \mathrm{V}_\mathrm{y}$ etc. are nodes in the graph, and$\mathrm{L}_{ij}\mathrm{L}_{xy}$ etc. are labels associated with edges.

then

$$V = (ID_i, Metadata_i, Owner_i, NftLabel_i) \mid i \in I$$

where I is a collection of indexes to distinguish different NFTs. each $ID_i$ is unique and represents the identification address of the NFT.

The adjacency matrix A can be expressed as

$$A_{i*j} = \begin{pmatrix} a_{11} & \cdots & a_{1i} \\ \vdots & \ddots & \vdots \\ a_{j1} & \cdots & a_{ij} \end{pmatrix}, i, j \in I$$

Add New Node

$$A_{[(i+1)*(j+1)]} = \begin{pmatrix} a_{11} & \cdots & a_{1(i+1)} \\ \vdots & \ddots & \vdots \\ a_{(j+1)1} & \cdots & a_{(i+1)(j+1)} \end{pmatrix}, i+1, j+1 \in I$$

included among these

$$a_{1(i+1)} = a_{(j+1)1} = a_{(i+1)(j+1)} = 0$$

Create a new relationship (create a one-way relationship r between nodes Vm,Vn)

$$A_{mn} = A_{mn} + 1, mn \in I$$

$$L_{mn} = L_{mn} \cup r, mn \in I$$

Verify relationship (verify one-way relationship r between Vm,Vn)

$$r \in L_{mn}, m \cdot n \in I \Leftrightarrow true$$

Get all edges

$$OutEdges(V_i) = (V_i, V_j), A_{ij} > 0$$

$$InEdges(V_i) = (V_i, V_j), A_{ji} > 0$$

Get all neighboring nodes

$$Nodes(V_i) = \{V_j, A_{ij} > 0 \parallel A_{ji} > 0\}$$

## Diagram structure maintenance

Proof of Stake (PoS) is a blockchain network consensus mechanism designed to replace the traditional Proof of Work (PoW). In PoS, nodes (verifiers) of a blockchain network participate in the network consensus process and block generation by holding and "pledging" a certain amount of cryptocurrency. Unlike PoW, which relies on computing power, PoS relies on the amount of coins held by the participants and how long they have been held.The main advantages of the PoS mechanism include greater energy efficiency and reduced risk of network centralization. The mechanism is suitable for different blockchain and cryptocurrency projects, especially those seeking a more environmentally friendly and scalable solution.

In this system, a consensus algorithm based on proof of interest is also used.

1. Coin-holding verification: nodes (verifiers) prove their contribution to the network by holding and locking a certain number of homogenized tokens.

2. Selecting the validator: the network randomly selects the validator to create a new block based on the node's coin holdings and the wait time since the last packing, weighted according to the weight.

3. Block creation: the selected validator validates the pending transactions and packages them into a new block.

4. Block validation: other validators check the new block to ensure that all transactions are valid and compliant.

5. Confirmation on the blockchain: once a new block is confirmed by a majority of the network's verifiers, it is added to the blockchain and becomes an immutable record.

6. Reward distribution: validators who participate in creating and validating blocks are rewarded by the network with a certain amount of

gas fees.

## Deal structure

Similar to the Bitcoin chain structure, the chain structure of the system will verify the relationship between the previous block and the new block to build the new chain.

1. Check for predecessor blocks: after the chain receives a block, it checks whether the previous block being referenced exists and is valid.

2. Check time range: check if the timestamp of the block is greater than the timestamp of the previous block being referenced and within the consensus time of the round.

3. Check Consensus Result: Checks the consensus result of the block and discards it if there is no consensus.

4. Execute the block transaction: make the state at the end of the previous block Status. make the transaction list of the block TxList, which contains multiple operations. Loop through the transactions from the TxList and enter a status transition function to execute if there are no errors. Make StatusFinal the status of the last transaction, but add the block reward paid to the miner. Check if the Merkle tree root of status StatusFinal is equal to the final status root provided in the block header. If it is equal, the block is valid, otherwise the block is invalid.

# Realization and assessment

## Traceability system design

In order to test the effectiveness of the current blockchain system's features, a specific application scenario can be set up.

Most of the current NFT traceability systems are based on simple smart contracts, which can only realize simple data storage, while the source data is stored in decentralized data storage like IPFS. Such a structure leads to the fact that the traceability items and the process of traceability can not be well described, and the relationship between them can not be described more accurately.

Set up two groups of smart contracts, both of which realize the binding and finding of the relationship between the traceability process data and the traceability items, and the specific relationship flow is shown in the following figure, the first group of smart contracts is written based on the smart contract language of Ethernet, and the second group is realized based on the smart contracts of this system.

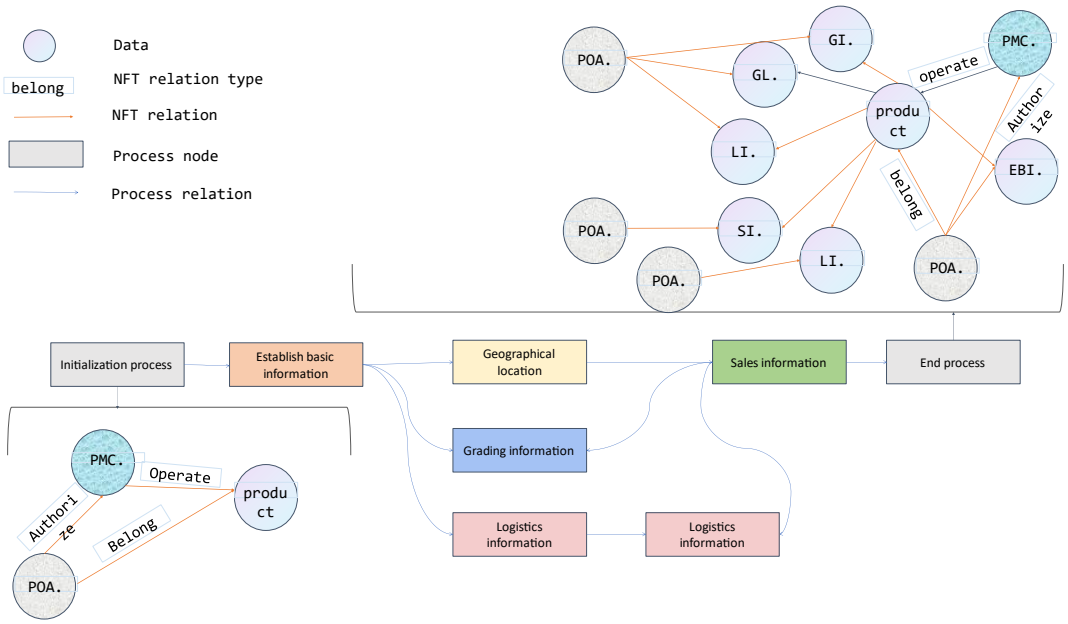


Chart 3 Schematic diagram of the traceability system

```
// Define an NFT class that can represent either a product or data
class NFT.
    def __init__(self, id, type, attributes).
        self.id = id
        self.type = type // 'product' or 'data'
        self.attributes = attributes // Attributes of the NFT, e.g., production date, origin, etc.
        self.relationships = [] // Relationships with other NFTs

// Define a Traceability Graph class to manage the NFTs and their relationships
class TraceabilityGraph.
    def __init__(self).
        self.nfts = {}

    // Add an NFT to the graph
```

```
    def add_nft(self, nft).
        self.nfts[nft.id] = nft

    // Establish a relationship between two NFTs
    def add_relationship(self, parent_nft_id, child_nft_id).
        if parent_nft_id in self.nfts and child_nft_id in self.nfts:
            self.nfts[parent_nft_id].relationships.append(child_nft_id)

// Define a function to perform the traceability process.
function PerformTraceabilityProcess(graph, product_id, data_attributes).
    // Create a new NFT for traceability data
    data_nft = new NFT(GenerateUniqueId(), 'data', data_attributes)

    // Add the data NFT to the graph
    graph.add_nft(data_nft)

    // Link the product NFT with the new data NFT
    graph.add_relationship(product_id, data_nft.id)

    // Return the updated graph
    return graph

// Example usage
// Initialize a new Traceability Graph
traceability_graph = new TraceabilityGraph()

// Create a product NFT and add it to the graph
```

```
product_nft = new NFT("product123", "product", {"name": "Apple", "origin": "Farm XYZ"})

traceability_graph.add_nft(product_nft)


// Perform the traceability process by adding data points throughout the product's lifecycle

traceability_graph = PerformTraceabilityProcess(traceability_graph, product_nft.id, {"date": "2024-01-27", "event": "Harvested"})

traceability_graph = PerformTraceabilityProcess(traceability_graph, product_nft.id, {"date": "2024-02-01", "event": "Shipped"})
```

## Environmental settings

In the experiments, both our proposed framework and Hardhat are deployed in Docker Engine 24.0, and in terms of consensus algorithm, since both of them use a consensus algorithm based on proof of interest, both are deployed only on a local single node in order to control the experimental variables.

## Data collection and analysis methods

During testing, our proposed TaYi framework uses tayijs to make connections and calls to the chain and web3js to make connections and calls to Hardhat, and the same code is written to record the response times before and after calls to both frameworks. Call the query class method and data change class method respectively and keep increasing the number of queries per second, record their response time and number of failures to approximate the execution time of the chain code with the request response time.

## statistical data

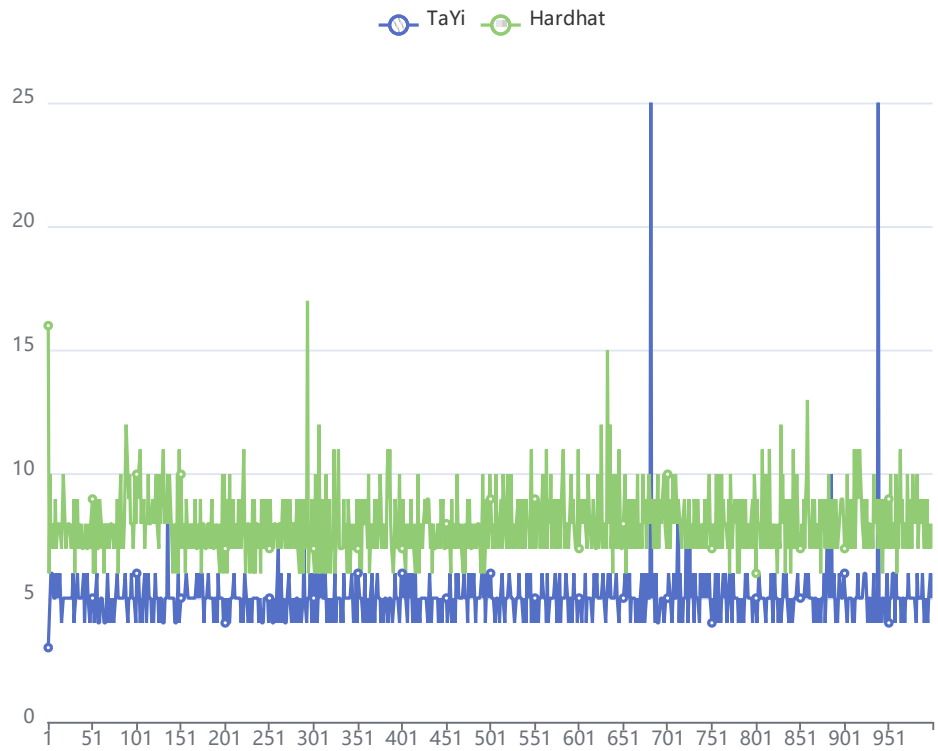


Figure 1 Single-threaded run of two systems' data separately (query operation)

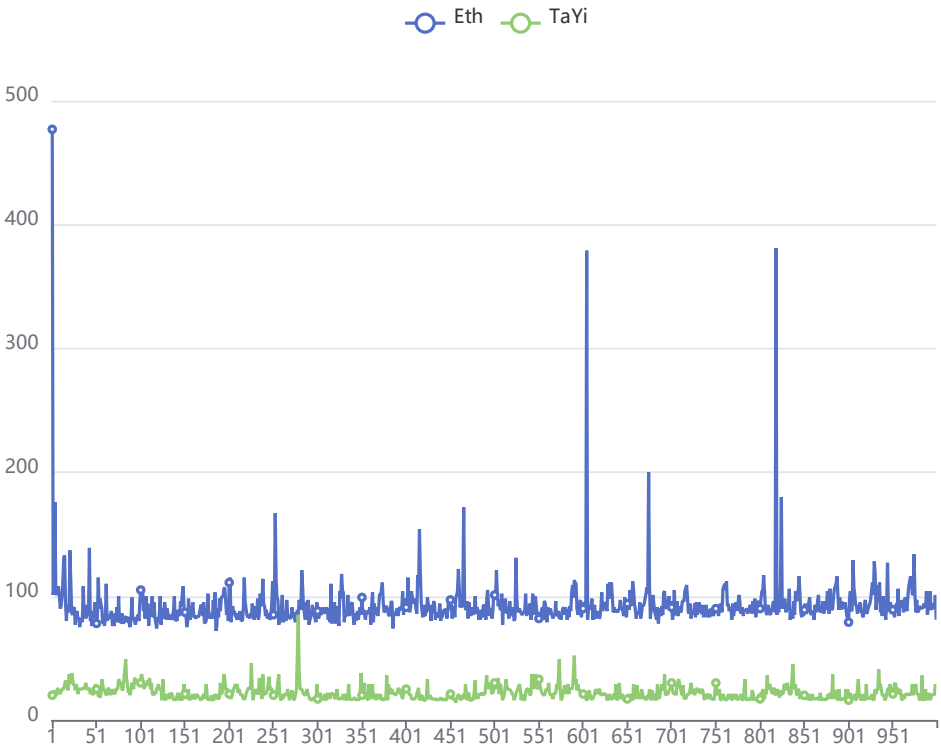


Figure 2 Single Threaded Write Data

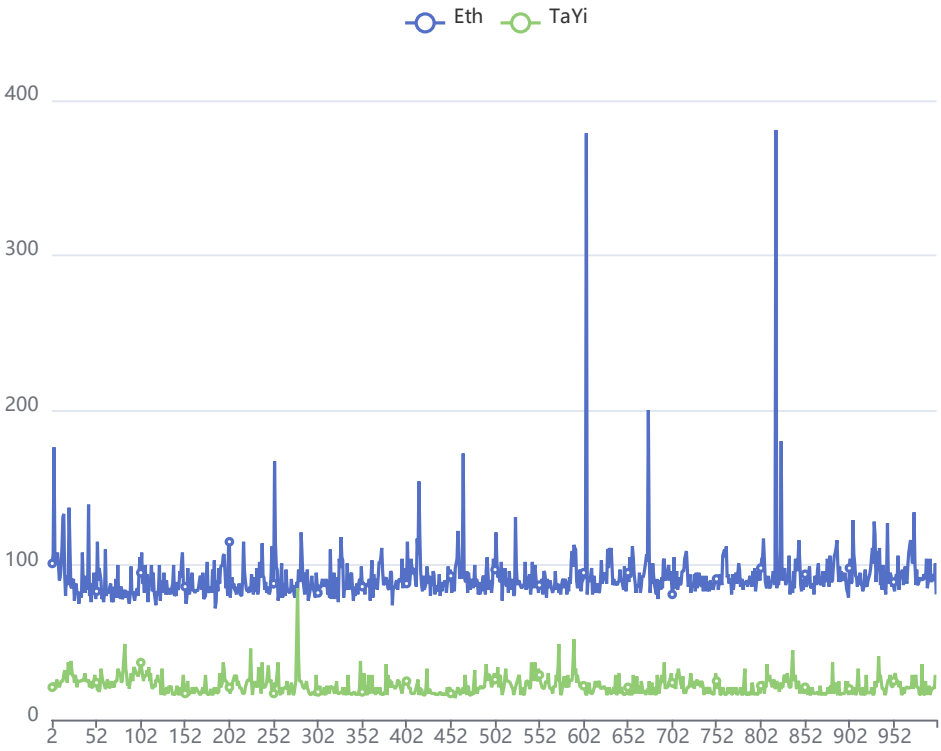

Figure 3 Single-threaded write data (with exception data removed)

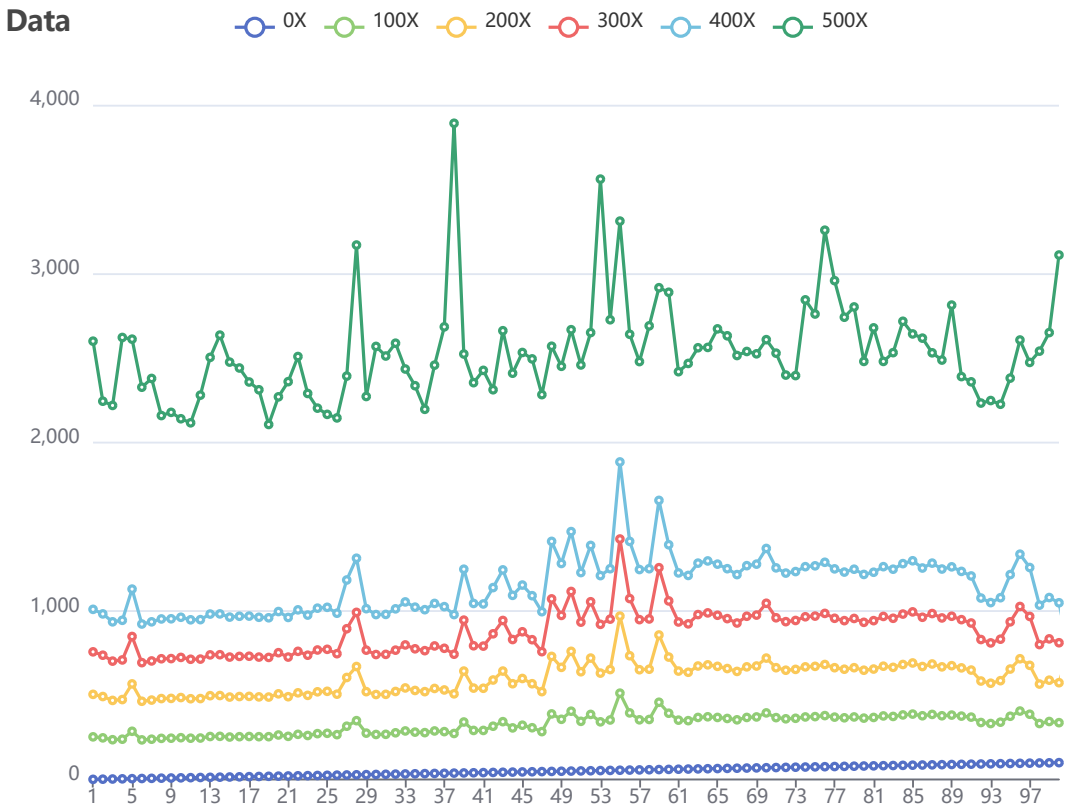


Figure 4 Relationship of data volume to query operation (Eth)

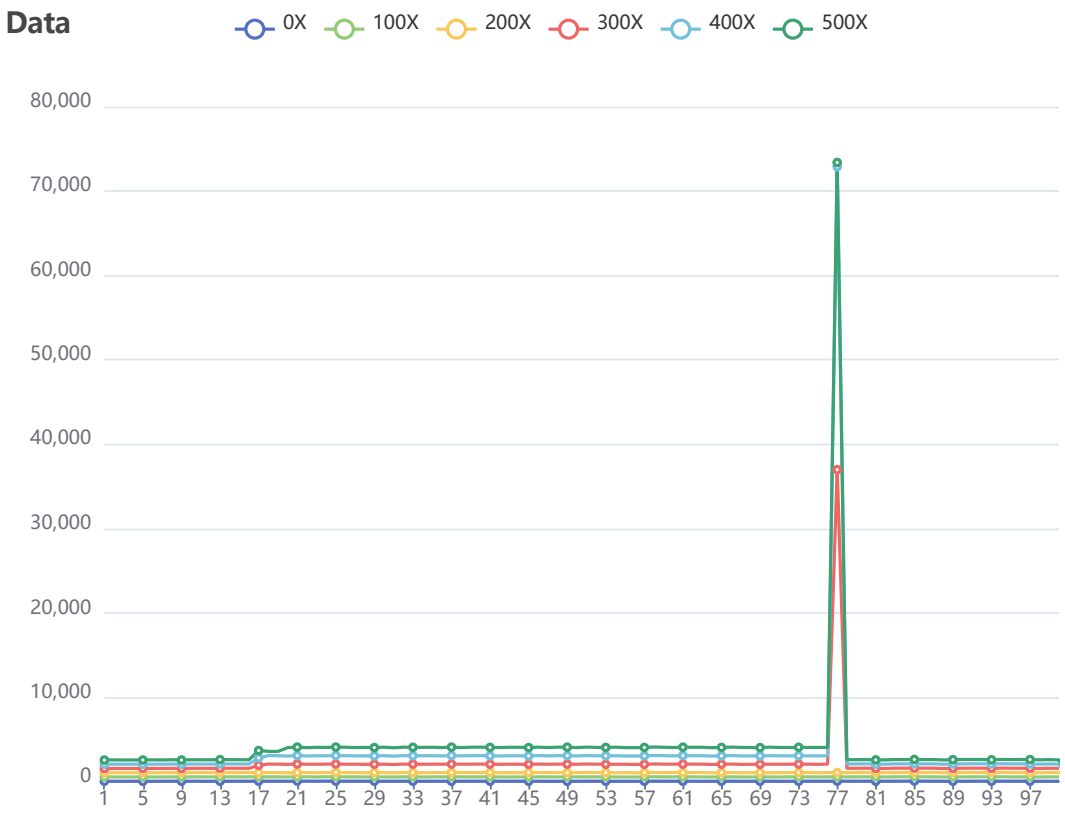


Figure 5 Relationship of Data Volume to Query Operation (TaYi)

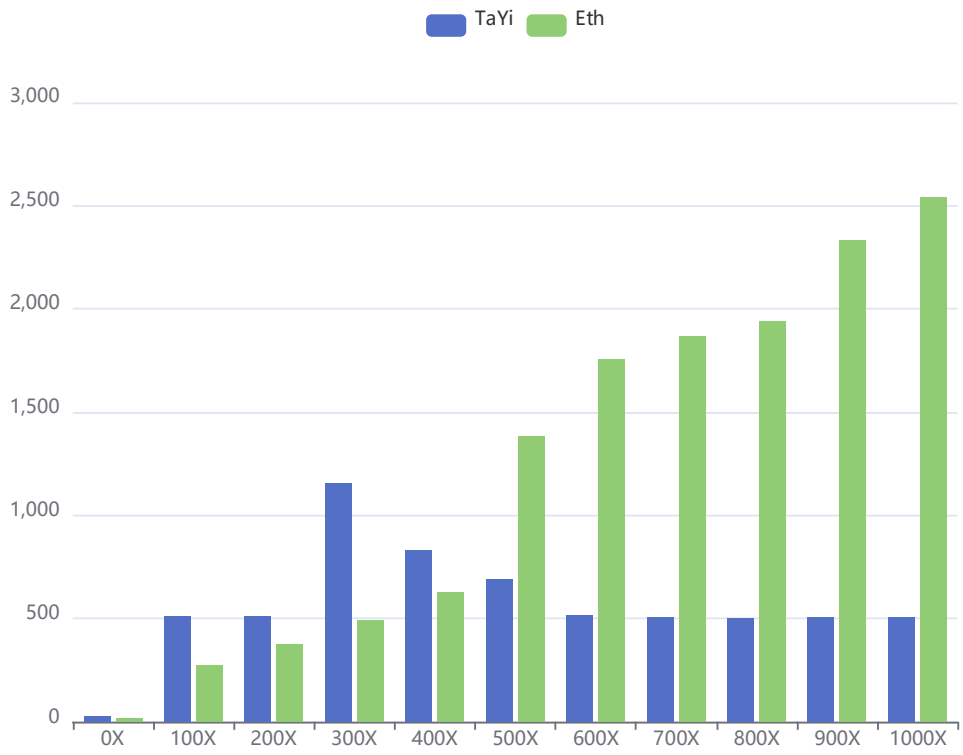

Figure 6 Comparison of the relationship of data volume to query operation

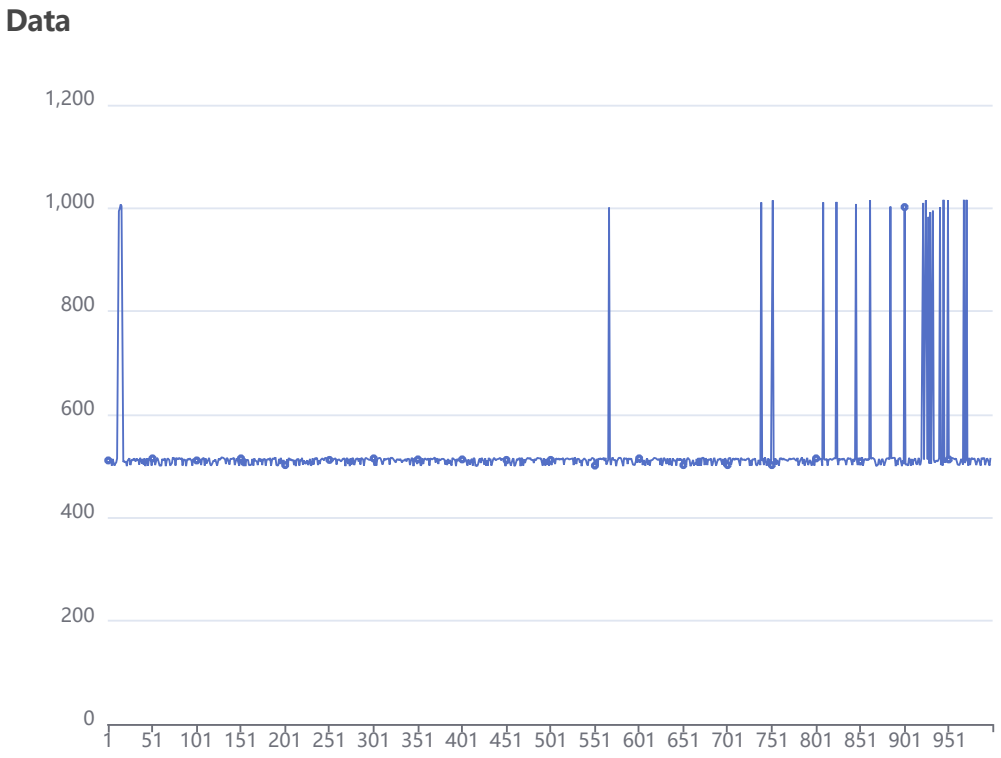


Figure 7 Relationship of Data Volume to Query Operations (10W)

# Talk over

In this study, we designed a blockchain based on NFT structure and NFT relationship graph structure, and deployed a single node TaYi in the test environment and a single node eth built by hardhat to compare the experiments using real cases. In this experiment, the performance of the two systems in terms of on-chain data query, on-chain NFT creation, on-chain NFT query is tested and analyzed.

There is a huge difference between the two systems in the performance of on-chain data querying, in the 1000 experiments of querying, the query response time of both are under 25ms, but in most of the tests, the query response time of TaYi is much lower than that of Eth. In terms of the on-chain NFT creation, it can be seen from the results of the statistics that in the first time of Eth's creation, it is possible that due to the initialization of the operation, which leads to this data abnormal, so the abnormal data can be removed for analysis. After removing the abnormal data, it can be seen that TaYi's performance is also ahead of Eth, and it can be seen that when the number of times is greater than 600, the length of time for Eth's NFT creation fluctuates greatly, compared with TaYi, which is very stable.

When exploring the relationship between the amount of NFT data that the system already has on the response time of the system's query operation, we can find that Eth has a large time variation when the system's data is 2,000 as a cutoff point, indicating that Eth at 2,000 times may be the critical point for this system when it comes to NFT queries. In the TaYi system, it can be seen that when the number of operations is 77 times, there exists an abnormal data due to unknown reasons. Except for the abnormal data, this system is running in a relatively smooth state, and the response time is basically under

1000ms. The average value of the operation in each data volume can be taken and compared with the data volume of the operation as a bar chart to find that, with 500 as the cut-off point, the performance comparison has a huge change, and after 500, the response time of Eth increases rapidly with the increase of data volume, but TaYi is still very stable. In order to explore the data threshold of TaYi, the arithmetic mean can be taken by testing 1000 times for every 100 data volume increase. From the experimental results, it can be seen that the data fluctuation of TaYi is getting bigger and bigger after the data volume of 90000, which can be considered as the critical point of TaYi.

# Reach a verdict

In this paper, with the data reliability and ownership characteristics of NFT, inspired by the ERC721 and EIP6551 standards, we abstract a new NFT on-chain expression and reconstruct the blockchain state transition system based on this expression, realizing an NFT structure based on the NFT structure that can be efficiently described and queried by the NFT relationship graph structure.

The system provides a powerful smart contract relationship management interface, which breaks through the limitations of traditional blockchain in the description of complex scenarios compared to traditional blockchain. It is especially suitable for scenarios that require efficient description of complex relationships, such as social networks, industry chain tracking, and authorized data management.

In addition, the system's privatization smart contract, which combines the ownership characteristics of data accounts, provides new possibilities for privacy protection. Especially in the field of sensitive information processing, such as medical data and business data, the system can provide a more secure and private data processing and storage solution. The design of this system not only improves the efficiency and security of data processing, but also opens up new paths for the application of blockchain technology in a wider range of fields.

After an experimental comparison with Hardhat, in the single node case, the performance of the system is more indeed superior to Eth in certain scenarios, especially in terms of on-chain NFT operations, which outperforms Eth at 10W data volume.Our findings provide a deeper understanding of the construction of blockchains for specialized domains and the value attributes of NFTs. This is particularly evident in the specific application domain of the TaYi system. However, our

study has some limitations in terms of larger data writing and management, and the existing systems are inadequate for handling very large data under the trend of increasing data, which may affect the industrialization and large-scale application of such systems.

In our future work, we aim to improve the value and usability of authorized data on the basis of ensuring data privacy and security. The superior data protection and data retrieval functions of the current system can provide new ideas for data authorization for AI, which can authorize the required data for AI and promote the development of AI in the chain while ensuring data security. At the same time, data and identity can be shared based on the system, and virtual and real property can be combined to build a more perfect open world on the chain.

In summary, although our TaYi system still needs to be optimized in handling ultra-large-scale data, the prospects it shows in the application fields of NFT and blockchain technology are already obvious. Through continuous technological innovation and improvement, we believe that in the future TaYi system will not only be able to overcome the existing challenges, but also open up new paths for the diversified applications and comprehensive development of blockchain technology, thus bringing more far-reaching impact and value to our digital world.