QITCOIN

WHITEPAPER

QITCHAIN NETWORK

Qitchain Network

Overview

The QTC Blockchain Database System for Rich Query

As a distributed ledger jointly maintained by multiple parties that do not trust each other, the data query service provided by the blockchain system must meet the requirements that the query result set has not been tampered with and the integrity can be verified.

The query in the blockchain system is mainly divided into two categories:

1. Queries oriented to full nodes
2. Queries oriented to light nodes

All queries require the full node to maintain a complete blockchain database, participate in consensus, and synchronize the data. Typically, the storage, calculation, and network costs are high.

While performing light node queries, users only need to maintain the block. Header information is used to verify the query result set, which was returned from other full nodes. As a result, costs are significantly reduced. Most of the existing blockchain systems store data in key-value databases or file systems with simple semantic descriptions, such as level DB. However, this type of system has a single query interface, and the types of queries that it can support for full nodes are limited. Most of the current work provides more query services by copying the block data to the off-chain database. Still, the additional storage of the block data dramatically increases the storage cost.

For light node queries, the inquirer needs to verify the integrity of the query result set from untrusted full nodes. However, traditional blockchain systems only support simple transactions or state existence proofs and cannot verify other types. Moreover, the MB tree-based verifiable query scheme widely used in outsourced databases is challenging to adapt to the update characteristics of block data, resulting in underperformance. For this reason, it cannot be directly applied to light node-oriented queries.

In light of the problems mentioned above in query processing in our current blockchain systems, this document outlines and implements a new blockchain database system with rich queries. That is the Qitchain mining opportunities on the public chain.

Chapter 1: The origin of Qitchain

A) Query processing for full nodes

As blockchain applications grow in scope and volume, the demand for querying these systems increases as well. Although the full node stores all the block data, the current blockchain system has limited support for full node queries, and it still has the following shortcomings in data query processing:

1. Data semantics are not rich enough. Blockchain systems mostly use key-value models with limited expressive capabilities, and the semantic description of transaction data is insufficient, making it challenging to support complex queries.
2. There is insufficient data manipulation. Existing systems usually store block data in a key-value database or file system, both of which only support simple access methods. Therefore, although the transaction is structured, the existing system cannot support the relationship between transactional data and operating data. Some systems import block data into off-chain databases for query, but data migration brings additional system overhead, and data replication and storage bring additional storage costs.
3. On-chain/off-chain data integration is complex. Existing solutions store large-scale data or private data in an off-chain database, obstructing the blockchain. Storing as such reduces network overhead and protects privacy. Additionally, this method makes the information related to the same entity be stored on the chain and off the chain simultaneously, burdening fruitful data integration. Due to data consistency and data privacy issues, users cannot use smart contracts to access off-chain data. Importing blockchain data into off-chain databases for queries will introduce additional storage overhead.

Relational semantics added

The lack of data semantics makes it difficult for blockchain systems to support complex queries. Data semantics must be added to support rich queries on block data. Most transactions are structured and contain both system and user-defined attributes. A couple of examples of system attributes are transaction hash and transaction sender. User-defined features include smart contract parameters.

Transactions that call for similar operations are structurally identical. Based on this, Qitchain adds relational semantics to the block data. A unified relational model describes the same type of transaction, and the transaction structure is determined by declaring the relational table model. The attribute types are either characters, numbers, or other types. The two categories of table attributes are system and application attributes. System attributes are automatically added when the table mode is created. Two examples include Tname (transaction type) and SenID (transaction sender). The user explicitly defines the parameters, and they can send a unique transaction to create a table pattern, which can be synchronized between nodes through a consensus algorithm.

For example, in Figure 1.1, we can see donation, transfer, and distribution transaction types. Users declare the relationship tables per transaction. In this case, the donation table consists of donors, projects and amounts. The transfer table relates projects, donors, organizations, and quantities. Distribution tables work with projects, donors, organizations, domes, and quantities. The data of each table is composed of corresponding transactions in the blockchain, and the transactions logically belonging to the same table are physically distributed in different blocks.

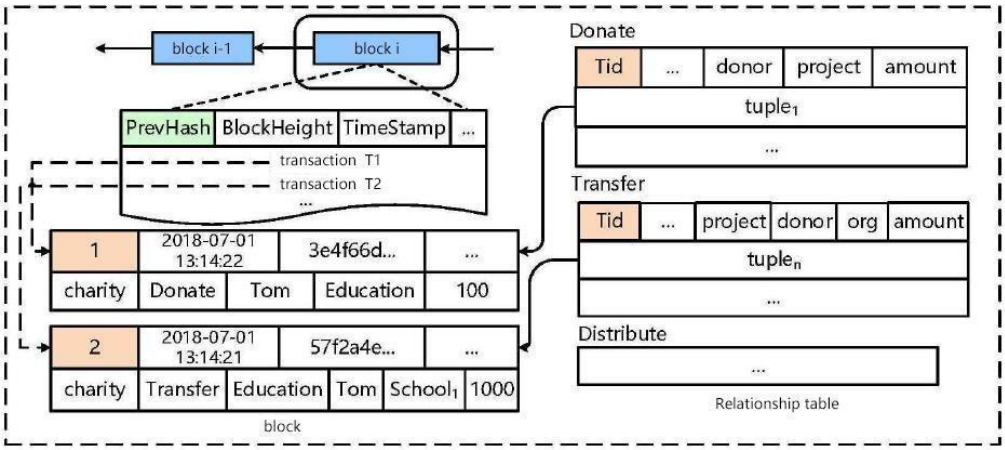

Figure 1.1 Application case of charitable donations.

Query definition

Definition figure 1.2 shows a case of on-chain/off-chain integrated deployment. In practical applications, data is usually stored in both the blockchain system and commercial relational database management system (RDBMS), such as MySQL. According to the block structure, data needing to be shared among these applications is stored in the blockchain, known as on-chain. Frequently private, large-scale data is managed by the local RDBMS, known as off-chain. Then there is data under the chain, which is stored in the relational table.

These distinctions lay the framework for three different query tasks:

1. On-chain queries
2. Off-chain queries
3. Joint queries (both on-chain and off-chain simultaneously)

Qitchain focuses on the first and third because the local database is all one needs for query number two.

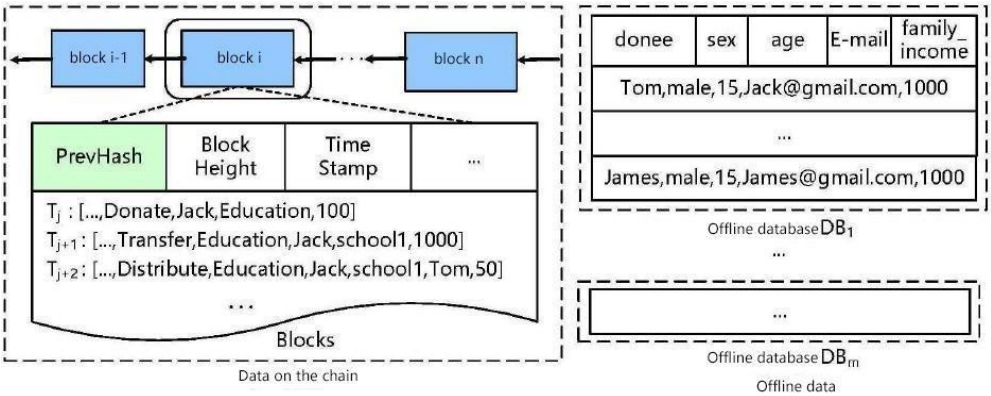

Figure 1.2 A case of on-chain/off-chain integrated deployment.

The types of queries

1. On-chain queries these are divided into three subcategories:
    a. Block and transaction query (common place in blockchain)

b. Retrospective query (supports historical transaction tracing)
c. Relational query (relational semantics)

2. Block and transaction query

The basic query in blockchain. Quite simply, block queries return the corresponding block, and transaction queries return the transaction information for the corresponding number.

3. Retrospective query

Traceability is of utmost importance in the blockchain ecosystem, most notably in decentralized evidence storage. Tracing historical transactions is two-dimensional, encompassing both sender and transaction type. From these dimensions stems three distinct operations a given transaction sender, a given transaction type, or a given transaction sender and transaction type.

4. On-chain relationship query

Since the relational model describes the block data, the relational query in the traditional relational database is also suitable for the blockchain system. Users can query all block data through a relational query, and through a specified time window, they can also obtain query results within a specified period.

5. Joint query

Data about the same entity is stored in the on-chain/off-chain databases simultaneously. Its integration is supported through operations and off the chain, called joint queries. For example, in the query "to obtain the family information of a certain recipient of a certain project," the query result can be obtained by connecting the tables on and under the chain.

Research in this chapter has brought forth the following conclusions

1. Insufficient blockchain data semantics cause limitations in full-node queries. By introducing relational semantics in the blockchain ecosystem, complex queries are enable, composed of basic relational operators such as selection, projection, and connection. As a result, query processing capacity improves even though only one piece of data is stored. In addition, additional costs are avoided.
2. In block storage mode, apparent problems are surrounding the low performance of relational queries. Table-level bitmap index and hierarchical index are designed to speed up data access. In addition, successful research has optimized some key relational and blockchain operators, such as retrospective and joint queries.
3. The above technologies are integrated into the independent blockchain database system Qitchain. To test the query performance of the blockchain system, a benchmark chain Bench is proposed to evaluate the data query performance. Experimental results show that index based query optimization can effectively improve query performance.

B) Verifiable range and connection query processing for light nodes

Since the query processing for full nodes cannot prove the integrity of the query result set, it cannot be directly used to process query requests from light nodes. Traditional blockchain systems mostly use merkle trees or other variants, such as a merkle patricia tree (MPT), which only support simple proof of existence, such as transaction verification or account balance queries. They cannot support the verification level achieved in outsourced databases regarding multiple query types, making it

challenging to satisfy the growing requirements of light nodes. Two additional challenges have been identified in blockchain in regards to verifiable Range and connection queries.

1. The verifiable query of the MB tree cannot be directly used in the blockchain due to low writing efficiency.
2. Data queries can only be made within a certain period on the blockchain. A write-efficient ADS AC tree is designed to optimize the query based on the time window in response to the aforementioned challenges. Based on AC tree And AC* tree, a verifiable connection query is realized, dramatically improving the query performance of the verifiable range query and connection query.

C) Verifiable aggregation query processing for light nodes

With the widespread application of blockchain systems, users' demands for data analysis on the data stored in the blockchain have gradually increased. The aggregate of queries paves the way for convenient data analysis. Therefore, supporting aggregate queries helps to improve the data analysis capability of the blockchain system in a simple way. This article mainly focuses on aggregate queries with multiple selection predicates, namely multidimensional aggregate queries.

Supporting verifiable multi-dimensional aggregation queries on the blockchain faces the following challenges:

ADS construction and storage overhead. It is necessary to build and store an ADS in the order of magnitude because traditional ADS's that support verifiable multi-dimensional aggregation queries, such as AAR trees Based on R* trees, have performances that diminish as dimensionality increases.

The variance of block data with traditional ADS, it is impossible to create all ADS's at one time to meet query requests of anyone dimension due to storage overhead. Users need to create ADS's to meet query requirements dynamically. However, this method needs to modify the historical block data to save the root node of the ADS in the block header. The historical block data cannot be changed, thus destroying the immutability of the block data.

In response to the above challenges, we researched and proposed a cryptographic accumulator based ADS GCA2 tree to support verifiable multi-dimensional aggregation queries for blockchain. For the first challenge, the GCA2 tree only needs to build a sorted list for each query dimension. Therefore, its construction and storage costs Increase linearly with the Increase of the query dimension. For the second challenge, the query processing based on the GCA2 tree transforms the verifiable multi-dimensional aggregate query into a nested set query. Thus, a single GCA2 tree can support multi-dimensional aggregate queries of any combination of dimensions.

Chapter 2: Qitchain - A blockchain database system for rich queries

A) System architecture overview

This part will introduce three aspects: network architecture, data model, and single point architecture.

QTC network Architecture

Figure 2.1 shows the network architecture of Qitchain. Multiple participants ally and each participant maintains a blockchain node, which is organized into a trusted alliance chain through the network. In this architecture, four types of nodes are included:

1. Certificate authority (CA):

An authority responsible for issuing and managing digital certificates, usually a trusted third party, responsible for verifying the legitimacy of public keys in the public key system. Qitchain provides an alternative CA to identify participants' identities in the current system.

2. Full nodes:

A full node is a node in the system that participates in consensus and saves complete data. It has high requirements for storage, computing, and network resources. Users who maintain a full node usually need to pay a relatively high maintenance cost.

3. Light nodes:

Individual users usually run light nodes. Only the block header data is stored without having to bear high system maintenance costs. The transaction or data query request needs to be forwarded to all nodes for processing.

4. Off-chain database:

Database Stores private data or large-scale data to protect data privacy and reduces system network overhead. By adjusting the configuration file, Qitchain can connect to different off-chain databases. Users can query the off-chain database individually or perform joint queries through Qitchain.

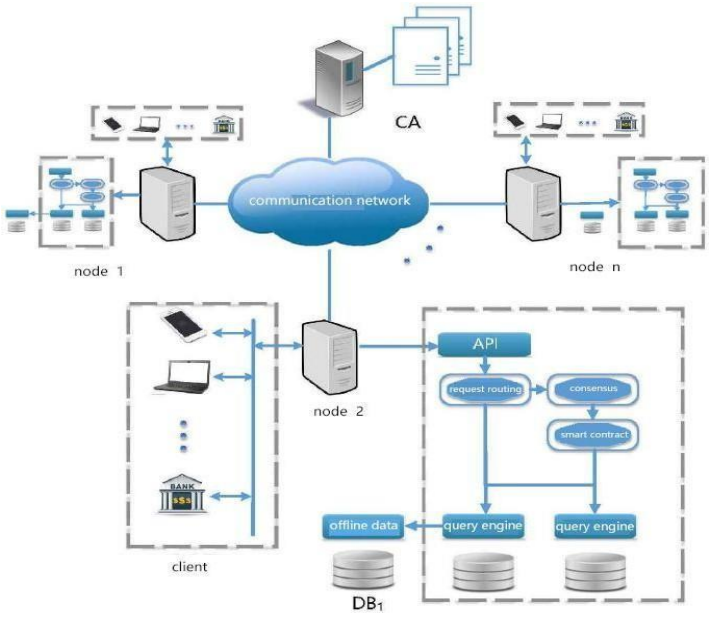

Figure 2.1 Qitchain

Data Model

Qitchain models block data through a relational model, and a unified relational model describes transactions of the same type. The attribute type can be characters, numbers, or other types. Attributes are divided into two categories: system-level attributes and application-level attributes. System-level attributes are automatically added when the table mode is created, such as transaction signatures and timestamps. The user and store user parameters define application-level attributes. There are two ways to create a table mode that Qitchain supports:

1. Create a blockchain table the user first declares the table mode, then inputs data by sending the transaction. The transaction does not need to execute the smart contract.
2. Create a table mode for transactions of the same type the user first deploys the smart contract. The transaction statement table mode of the contract calls the smart contract by sending a transaction, and the transaction data is also the table data. Blockchain lists are suitable for applications with simple transaction logic, such as data storage, eliminating the need for smart contracts, which maximizes performance. Instead, smart contracts are better suited for more complex business scenarios. The user creates a table mode by sending a unique transaction and synchronizes table mode information between nodes through a consensus protocol. Qitchain supports data management through SQL-like language. Users can use CREATE, INSERT and SELECT statements to create tables, send new transactions, and obtain query result sets. In addition, Qitchain can also easily define more query statements to support more query types. This suits various scenarios, such as TRACE statements to perform retrospective queries and JOIN statements to support simultaneous data queries from the same source.

Here are three statements for data manipulation in Qitchain:

a. CREATE Donate（donor string, project string, amount decimal)
b. INSERT into Donate ("JACK," "Education," 100)
c. SELECT * from Donate where donor = "JACK"

Adding relational semantics to block data enable Qitchain to hide complex system operations behind simple system Interfaces. Developers only need to use familiar SQL-like statements (such As INSERT, SELECT) to manage blockchain data.

Single point architecture

Figure 2.2 shows the system architecture of the Qitchain system, including the application layer, query layer, storage layer, consensus layer, and network layer.

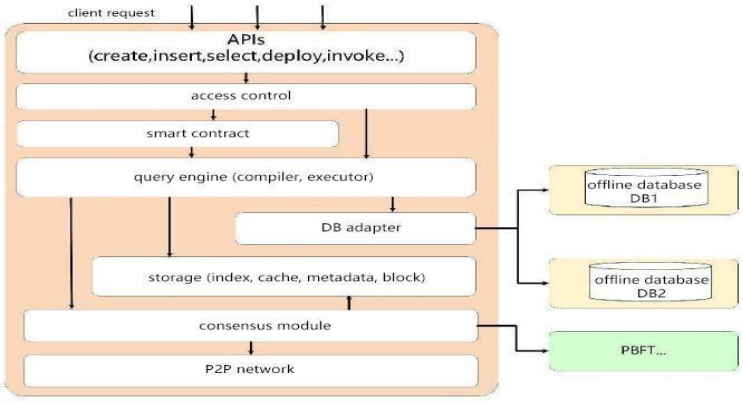

Figure 2.2 Qitchain system architecture

1. Application layer:

It consists of three modules:

a. API:

Provides a way to access the blockchain in SQL-like languages.

b. Permission control:

Verifies the requested permission. Qitchain protects user privacy through a multi-channel mechanism.

c. Smart contract:

Used to create DAPP (Decentralized Application).

2. Query layer:

Receives the client's query request, performs query analysis and query plan generation and execution, and returns the query result set to the client. The compiler is responsible for compiling the request into an internal representation, that is, a parse tree and the executor is responsible for generating and executing the query plan.

3. Storage layer:

Responsible for data structure maintenance and storage, that is index creation and maintenance, verification structure creation and maintenance, cache management, metadata maintenance, and block storage. In Qitchain, the storage layer mainly includes four modules:

a. Cache module
b. Index module
c. Metadata module
d. Block storage module

4. Consensus layer:

To ensure the consistency of the block data among the blockchain nodes, participants need to reach a consensus on the block data. The consensus layer guarantees data consistency between the nodes through a consensus protocol. Different consensus algorithms can be used in different application scenarios to satisfy various needs.

5. Network layer:

Responsible for message dissemination and data verification. Qitchain adopts the gossip protocol, which is widely used in fault detection and member detection of distributed databases and in many blockchain systems to realize block propagation and block data recovery.

The transaction execution process in Qitchain is as follows: users first deploy smart contracts or create blockchain lists on Qitchain through programming interfaces according to application requirements; then, for applications deployed using smart contracts, users can create table patterns for different transaction types to add relationships semantics; After that, users can update the blockchain by calling smart contracts or inserting data in the blockchain list by sending transactions. The transactions sent by users will be packaged into blocks according to the period or threshold of the number of transactions and consensus; finally, users can use SQL-like languages to query block data.

B) System implementation

The system implementation includes the storage layer, query layer, consensus layer, smart contract implementation, and system crash recovery.

Storage layer design

Qitchain maintains both on-chain and off-chain data. The off-chain data is managed by the local RDBMS and provides access functions through the data access interface. This section focuses on on-chain data. The storage layer is responsible for data storage and index maintenance and includes four sub-modules:

1. Block storage

Module responsible for the storages and recovery of block data and disk space 16 management.

2. Metadata module

Responsible for the storages and recovery of metadata.

3. Cache module

Maintains and updates the transaction cache. Qitchain uses the LRU cache mechanism to reduce disk READ overhead.

4. Index module

Responsible for the creation, maintenance, and restoration of the index structure and verification structure.

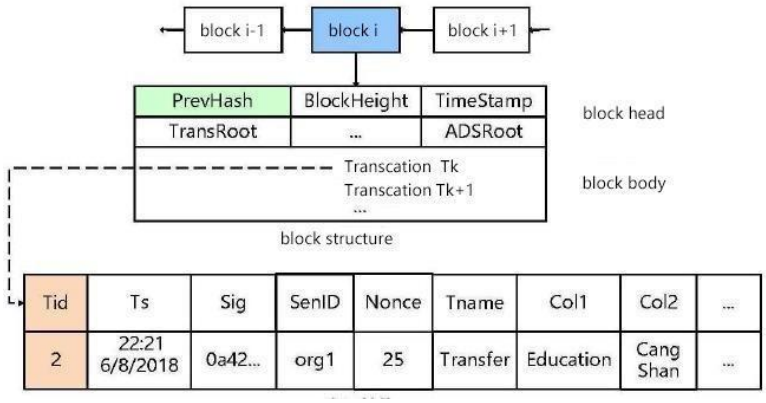

Figure 2.3 Block structure

The block consists of a header and body. The header records the meta information of the block. 'Prevhash' is the hash value of the previous block, block height is the current block height, timestamp is the block packaging timestamp, TRANS Root is the root node of the merkle tree established on the transaction, and ADS Root is the hash value of the root node of the merkle tree established on the root node of the verification structure.

The block body comprises a set of transactions, and each transaction contains system attributes and custom attributes. Tid is the 17 Transaction number, an increasing integer. Ts records the time when the transaction is packaged. Sig is the signature of the transaction sender. Send records the identity of the sender of the transaction. Tname indicates the transaction type and the relationship table to which the transaction belongs logically. Nonce is an incremental integer designated by the client to prevent replay attacks. For example, from figure 2.3, let's look At Tid=2. This transaction indicates the organization org1 transferred funds from education to Changsha at 22:21 on June 8, 2018.

The goal of block storage is to use the storage space and parallel query capabilities under the multi-disk architecture. The disk space management module counts the capacity information of each disk and the information of the block data volume. When storing a new block, the disk management module allocates disks for the new block according to the current data volume on each disk. The current storage data volume is compared, and the smaller disk has a higher priority. If the disk with the highest priority is full, the disk next in the priority pecking order is selected to store data. The block data is evenly stored on multiple disks through the above method, ensuring space and query capabilities are fully utilized. Blocks are stored as files. The file size of the storage block will affect the performance of the system. If the file capacity is set too small, there will be a large number of small files in the system, which will bring additional overhead to the file system. If the setting is too large, the query may be concentrated on a specific disk or a few disks, and the advantages of multiple disks cannot be used. The default file size in Qitchain is 256MB, and users can adjust the default file size through the configuration file to obtain better performance.

Metadata in the system includes table metadata and index metadata. Table metadata records information such as table mode, and index metadata records information such as index types and index columns. Due to frequent access, metadata is maintained in memory to improve metadata query efficiency. Table metadata and index metadata are organized in a sorted list according to table name and index name, and target data can be quickly located by binary search. To avoid the need to scan the block from the beginning to obtain the table or index metadata, the metadata needs to be synchronized to the disk for persistent storage when the downtime is restored. There are two solutions for storing the metadata to the disk:

1. Regularly store the data in the memory metadata information on the disk. In this way, the sorted meta information can be read directly from the disk when the crash is restored, reducing the recovery time.
2. Adding table and index creation transactions in the block. The method is stored in the storage files of table metadata and index metadata, and the file is read during the downtime recovery and reorganized once it is reorganized. This method takes longer to recover but reduces the cost of synchronizing metadata during system operation.

To make full use of the temporal and spatial locality, Qitchain has designed a caching layer on top of disk storage. There are two granular queries; block and transaction. An example of a block query is obtaining a block of a certain height. An example of a transaction query relates specifically to a unique transact.

Qitchain also supports relational queries, so most are transitionally granular. The use of block cache will bring about a couple of problems. First, memory space is wasted; often, only part of the query results in a block that needs to be stored in memory. Second, memory copy overhead is significant, and the blocks are usually large, so the memory copy overhead is also relatively large. Qitchain tackles these problems through the maintenance of an LRU-based transaction cache to improve query performance.

To optimize the query performance for full nodes, Qitchain supports creating three indexes: block index, table-level bitmap index, and hierarchical index. Ordinary index creation does not need to be synchronized to other nodes, that is, index construction and local index update operation. To support verifiable queries for light nodes, Qitchain endorses the creation of verification structures. The design and maintenance of verification structures require consensus to record all verification structures' root node hash values—Qitchain stores these root node hash values in a merkle tree. The hash value of the root node of the merkle tree is stored in the ADSROOT field in the block header.

Query layer design

Qitchain supports two query types: full node-oriented query and light node-oriented query.

1. Full node query:

Qitchain can support relational queries on block data by adding relational semantics to block data. Query processing for full nodes in Qitchain is similar to traditional RDBMS in the execution of analysis, logical plan generation, and physical plan generation. Qitchain first uses a query compiler to parse the query and generate an abstract syntax tree. Next, the syntax tree is converted into a logical plan and performs some basic logical optimizations. Finally, a physical plan is generated and executed.

In addition to relational query, Qitchain also customizes some special blockchain query operations, such as retrospective query and joint query. A joint query can realize the integration of on-chain/off-chain data. The user specifies the type of the off-chain database through the configuration file. When performing a joint query, the query module first converts off-chain data according to the kind of database. Then, the module performs the connection operation between the on-chain table and the chain list. The user needs only to make slight changes to increase the database types supported by Qitchain.

2. Light node query:

Since the accuracy and totality of the query set cannot be guaranteed, the full-node query process is challenging to use verifiably. To stride towards improvement, first the logical and physical plans are determined. Qitchain uses the verifiable query algorithm as previously outlined to support the query for light nodes.

The query verification structure needs to be constructed in real-time in the query execution process, so the coupling between the verification query and the storage layer is deep. After the query module receives the request, it will first determine if it is verifiable. If it is, it will be processed separately according to the verifiable query algorithm. If it is a general query, it will perform general query processing.

Consensus layer design

In Qitchain, the consensus module receives the transaction request sent by the client, packs it into a block, and runs a consensus protocol to reach an agreement on the block data among multiple nodes. Different consensus protocols can be applied in different application scenarios depending on the various needs. Blockchain ecosystems typically utilize consensus protocols based on computing power, such as Proof of Work (POW), preventing stoybil attacks. However, this method is not energy efficient. Delegated Proof of Stake (Dopes) improves this inefficiency. Other systems use PBFT, communication -based protocols are used as a consensus mechanism to improve consensus performance. If it is assumed that the nodes in the system are not malicious, then non-byzantine fault-tolerant consensus protocols such as raft or paxos can be used to improve performance further. Qitchain adopts a pluggable consensus module, allowing users to choose different consensus protocols to perform various application scenarios better. User requests are organized into a transaction form. Then, according to the consensus algorithm running in the current system, the transaction is packaged for consensus. The package operation is triggered by setting the timeout period and the number of transactions threshold. Qitchain currently supports Kafka and PBFT formula mechanisms. Under the Kafka consensus mechanism, user-initiated transactions are sorted through Kafka, and nodes consume messages (transactions) in a consistent order from Kafka, ensuring the consistency of transaction order among multiple nodes. Under the PBFT consensus mechanism, after the master node receives a certain

number of transactions or the timeout period is reached, the transaction is packaged into blocks, and the consensus process is initiated. After a consensus is reached, all correct nodes will receive the same block.

Smart contract layer design

1. Implementation of smart contract

The blockchain system can support the realization of more complex business logic with the help of smart contracts. By deploying smart contracts in the system, trusted transaction logic can be executed without a third party. Almost all blockchain systems have built-in smart contracts to implement complex transaction logic. Qitchain implements smart contracts that support Turing's complete language based on Docker.

2. Status data

To implement rich operations in smart contracts, Qitchain supports the use of couch DB to store smart contract status data. The data stored in the state database is organized in an MPT tree. The hash value of the root node of the MPT tree is stored in the block header. The MPT tree supports the light node's availability of state data. The hash value of the root node of the MPT tree stored in each block header corresponds to a snapshot of the state database, which can support fast restart and recovery of nodes.

3. Downtime recovery

Due to hardware errors or attacks, the node may be down during the service process. After the node is restarted, the system needs to be restored to the state before the downtime. The system variable 'curr_blockno' is used to record the system's block information and can be used to recover. When processing the i^{th} block, Qitchain first sets the value of the system variable 'curr_blockno' to i. Assuming that it is down while processing the i^{th} block, Qitchain downtime recovery needs to deal with the following situations:

a. Block data recovery

Keeps the block data of the node consistent with the latest block data in the system. In the gossip protocol, a node periodically Interacts with the node with which it is connected. If it finds that its data lags behind other nodes, it will automatically synchronize the data.

b. Metadata recovery

Restores the metadata to the latest state of the current node. The table creation transactions and index creation transactions in the block are stored in the corresponding metadata files. The metadata information before the block has persisted to the disk. When the system is restored, first read the metadata file to restore the metadata. Then restart the table creation and index creation transactions from the 'curr_blockno' block.

c. Index recovery

Restores the index structure to the correct state based on the block data in the current system. The table-level bitmap index is stored in the memory. To support the fast recovery of the table-level index, Qitchain supports the persistent storage of the table-level bitmap index to disk regularly. If the index is persisted to the disk, the table level bitmap index on the current disk is identified by the persistent version number n. The processing of hierarchical indexes and GCA2 trees is similar. The persistent version number is used to determine the current persistent index part and restore it from the block after the persistent version. When the block index is restored, the block information after 'curr_blockno' needs to be re-inserted. The restoration process of verifying the index AC tree and AC* tree needs to rebuild the C0 part according to the current state of C1. The smart contract performs crash recovery to

ensure the accuracy of contract execution. First, it ensures the atomicity of the transaction execution of the calling contract. There is no partial execution, it is either successful, or it was not, nothing in between. Second, it is guaranteed that all transactions in the current system are executed, and each transaction is completed only once. The recovery process of the smart contract during the downtime 24 recovery follows these steps:

i. Obtain the block number currently being processed (curr_blockno).
ii. Obtain the root node hash value of the MPT stored in the block header with the block number (curr_blockno-1).
iii. According to the above hash value, re-execute all transactions in the block with the current block number (curr_blockno). Reprocessing all transactions in the block ensures that all transactions are executed, and each transaction is completed only once.

Chapter 3: From mining to a decentralized search engine

A) QTC distribution and mining consensus algorithm

Based on the Proof of Capacity (PoC) consensus algorithm, QTC guarantees the healthy development of the entire cryptocurrency by designing a long-term incentive economic model. At the same time, it also improves upon PoC, making it a Conditioned Proof of Capacity (CPoC) consensus mechanism. Running congruently with Point of Stake (PoS), equity incentives have been increased, resulting in enhanced community consensus.

Total Supply	105 million tokens (105,000,000)
Qitcoin Foundation 5%	Long-term community construction, management, and other expenses lifted on January 1 every year beginning in 2022 for a total of five years.
Search Lab 15%	Research and development (R&D) expenses gradually lifted over five years Lift-off completion upon Qitchain becoming open-sourced on GitHub.
POS Rights 80% Of Miners	Storage miners and POS pledge TOP10 jointly obtain block rewards. 20% of the mining rewards obtained by miners are released at one time, and 80% rewards are released at 180 antennas. All reward maturity cycles are 100 blocks. Allocation: Miner mortgage: miner*80% POS: 20% Unsecured miner: miner*5% POS: 95%
Block Time	3 minutes
Initial Block Size	75QTC / Block, 2MB block size
Halving Cycle	The first halving time is 420,000 block height, and In the future, the height of 700,000 blocks will be halved every time
Initial Transactions Per Second (TPS)	70 TPS

Conditional Proof of Capacity (CPOC)	A pledge period of 360 days requires 10QTC/T, and a pledge period of 540 days requires 5QTC/T. The pledge demand halving is synchronized with the block halving period.

The issuance method of CPoC mining will positively affect miners, mining pools, foundations, and other parties, ensuring the entire system will always have a more dominant temporary commercial vested interest to promote the whole ecosystem invisibly.

B) QTC co-constructs a shared economic model

QTC's consensus algorithm has been upgraded based on burst PoC2 and named Conditioned Proof of Capacity (CPoC). This upgrade solves the following problems:

Prevent economic model attacks

The miners under the PoW consensus algorithm are forced to sell currency due to costs, which will lead to the shrinking of the entire mining economy. Miners in a CPoC's model become a part of a community in the interest of the whole ecosystem. In this model, the currency is instead used to replace the original electricity consumption resources. The ultimate positive benefit to the QTC ecosystem is continuous and automatic expansion.

PoW's high maintenance cost

Security maintenance requires high energy consumption. During market downtimes, electricity costs exceed mining rewards, forcing miners to sell tokens to pay for wasted electricity. Energy consumption value is not re-deposited into the currency system, so PoW mechanisms always lose value in this regard.

C) Product development is hindered through a lack of economic momentum

Without economic power, key technologies cannot be updated. As a result, long-term and effective development and iteration cannot be obtained, perhaps even causing subsequent team versions to forks undistinguishable from the main chain.

D) Mining machine monopoly

The PoW consensus algorithm will inevitably lead to an arms race in mining machines. To obtain higher computing power, special mining machines with higher performance will inevitably be developed, pushing ordinary people out of the market. With the PoC consensus algorithm, due to the slow iteration speed of hard disk manufacturers and the low threshold, there is no need to worry about the inability to purchase hard disks. As long as hard discs are available in one's country, anyone can participate in mining. In traditional commercial supply chains, suppliers generally do not become direct competitors of users. Still, ASIC manufacturers themselves in PoW are their largest miners, which means that ASIC manufacturers are direct competitors of miners and suppliers of miners. When the source of your business tools comes from your competitors, the profit you have is the risk part of being arbitraged by the other party, and the miners are entirely reduced to ASIC vendors' arbitrage tools.

E) Power resource monopoly

Hinders expansion of the PoW endogenous economic system. Revenue and expenditures are imbalanced, and mining costs exceed gains. For CPoC mining, the hard disk power consumption is low, and the miners' income will be more predictable. The linear hedging Rate of civilian computer hardware can also be used to ensure that miners can hedge against the risk of price fluctuations in the secondary market while maintaining relatively safe capital.

F) Mining process

1. P Disc (Plot):

The miner fills the hard disk with the hash value containing their public key and the integrated SHABAL algorithm on the local hard disk plot file. We regard the plot file (P Disk) as the process of software manufacturing (PoC mining machine) and release the power to monopolize the mining machine manufacturer to every ordinary person. The larger the hard disk capacity, the more hash value is filled, and the higher the probability of block explosion. The hash algorithm uses SHABAL256, which is resistant to ASIC.

2. Transfer (Transaction):

A peer-to-peer (P2P) network is composed of wallets based on Bitcoin (BTC) and transfer operations between them.

3. Forging:

The miner interacts with the P2P network through their wallet, and every time it receives a block, it starts the packaging process of the next block. The wallet organizes a block and sends the hash value to the miner, looking for the best matching nonce. After receiving nonce, the wallet converts nonce to deadline (time) and waits for the end of this time to broadcast the block.

4. Verify:

After receiving the block, verify.

Chapter 4: QTC Team

A) Founding team

1. Shoaib Hayat

Mr. Shoaib Hayat is an essential part of QTC and leading the team with his expertise in blockchain technology, where he is currently contributing to making QTC what it is. Mr. Shoaib Hayat is Computer Engineer with major of cryptography, network security, and possesses several decades of versatile experience related to blockchain technology and its applications. He has hands on experience on ethereal platform and implementing decentralized applications. He has worked with many multinational companies and contributed to them through his dedication, knowledge and vision.

2. Sam Catchpole

Mr. Sam Catchpole is Chief Technology Officer of Qitcoin, Mr. Catchpole has been working in information technology for 15 years. With extensive experience at Microsoft and later Google, his interest in blockchain lead him down an exciting rabbit hole of technology innovation. A creative programmer and technical framework expert, Sam brings creative concepts to Qitcoin to keep our technology at the forefront of the industry.

3. Yoshua Bengio

Mr. Yoshua Bengio is an outstanding member of QTC, and he is currently leading a radical change. A master of machine learning and deep learning. He received his PhD from McGill University. He is the initiator and researcher of appestat technology. He is also a tenured professor at the University of Montreal (Université de 30 Montréal). He has taught for more than 22 year. He is the head of the machine learning laboratory (MILA) and one of the leaders of the CIFAR project. He is responsible for neural computing and adaptive perceptron's. He is also the chairman of the Canadian society of statistical learning algorithms, and the chairman of NSERC-Ubisoft among others. Before teaching at the University of Montreal, he was a machine learning researcher at AT&T & MIT. His main contributions are in the fields of deep learning and artificial intelligence. He has extensive work experience, including technology transformation designed and managed by Google, data analysis and interpretation, and the development and implementation of AdSense research tools. He has held various leadership roles in major technology companies and now devotes all his knowledge and expertise to QTC.

4. Prasenjeet Kashyap

Mr. Prasenjeet Kashyap has been an international business and tech consultant for more than 10 years, with a varied background in multiple industries including supply chains, telecommunications and technology, Prasenjeet is an early tech advisor for IPFS projects, and one of the most sought-after IEO, IDO, STO, DeFi and IPFS experts in the blockchain and crypto space. Now he has sufficient background in IPFS-based storage solutions. As a member of the technical team, he has developed a strong learning curve that enable him to master web development, mobile development, cryptocurrency development and various cryptocurrency projects. Blockchain development. As the chief engineer and project manager of projects of all sizes, his experience enable him to understand the entire project and foresee possible problems in the future. Currently, he is dealing with internet security issues, which is an urgent problem in the world today.

5. Daniel Alejandro Lugo

Mr. Daniel Alejandro Lugo is a graphics designer specialized in corporate visual identity and social media design, with many years' of experience in both design and crypto, Daniel specializes increasing blockchain visual identity that created with passion, effort and dedication, using strategies and research to create designs that make sense and are relevant.

6. Daniel Ruvins

Mr. Daniel Ruvins is a highly regarded individual in the cryptocurrency space through his work with a multitude of successful start-ups over the past 5 years. His extensive experience shows in terms of business development, advisory services, and community management. His love for start-up environments and crypto allows him to thrive within the Qitcoin ecosystem and contribute greatly to its continued development.

7. Antoine Keusseyan

Mr. Antoine Keusseyan is a blockchain researcher with a degree in Business Management. A former member of the mining alliance, today he stands as the BD Director for Qitcoin, helping implement decentralization, information processing and blockchain technology into both day-to-day and business affairs.

B) QTC evangelists

Global blockchain faith evangelist the term "crypto economics" causes a lot of confusion. People are often unclear on what it is supposed to mean. QTC has a blockchain and crypto evangelist team to deliver industry depth and expertise to fast growth sector, working to spread knowledge and understanding of this technology on different forums. Our team has worked on many projects over the last decade. Our work has focused on prodding executives and entrepreneurs to think differently in their approaches to digital Darwinism and disruption. Whether we're speaking to startups about the next big thing, attempting to humanize corporate culture and innovation or exploring ways to engage audiences in more meaningful ways, art and story carry.