Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.047340

ARTICLE

Robust and Trustworthy Data Sharing Framework Leveraging On-Chain and Off-Chain Collaboration

Jinyang Yu[1,2], Xiao Zhang[1,2,3,*], Jinjiang Wang[1,2], Yuchen Zhang[1,2], Yulong Shi[1,2], Linxuan Su[1,2] and Leijie Zeng[1,2,*]

[1]School of Computer Science, Northwestern Polytechnical University, Xi'an, 710072, China

[2]MIIT Key Laboratory of Big Data Storage and Management, Northwestern Polytechnical University, Xi'an, 710072, China

[3]National Engineering Laboratory for Integrated Aero-Space-Ground-Ocean Big Data Application Technology, Northwestern Polytechnical University, Xi'an, 710072, China

*Corresponding Authors: Xiao Zhang. Email: zhangxiao@nwpu.edu.cn; Leijie Zeng. Email: zenglj@nwpu.edu.cn

Received: 02 November 2023 Accepted: 18 December 2023 Published: 27 February 2024

ABSTRACT

The proliferation of Internet of Things (IoT) systems has resulted in the generation of substantial data, presenting new challenges in reliable storage and trustworthy sharing. Conventional distributed storage systems are hindered by centralized management and lack traceability, while blockchain systems are limited by low capacity and high latency. To address these challenges, the present study investigates the reliable storage and trustworthy sharing of IoT data, and presents a novel system architecture that integrates on-chain and off-chain data manage systems. This architecture, integrating blockchain and distributed storage technologies, provides high-capacity, high-performance, traceable, and verifiable data storage and access. The on-chain system, built on Hyperledger Fabric, manages metadata, verification data, and permission information of the raw data. The off-chain system, implemented using IPFS Cluster, ensures the reliable storage and efficient access to massive files. A collaborative storage server is designed to integrate on-chain and off-chain operation interfaces, facilitating comprehensive data operations. We provide a unified access interface for user-friendly system interaction. Extensive testing validates the system's reliability and stable performance. The proposed approach significantly enhances storage capacity compared to standalone blockchain systems. Rigorous reliability tests consistently yield positive outcomes. With average upload and download throughputs of roughly 20 and 30 MB/s, respectively, the system's throughput surpasses the blockchain system by a factor of 4 to 18.

KEYWORDS

On-chain and off-chain collaboration; blockchain; distributed storage system; hyperledger fabric; IPFS cluster

1 Introduction

The rapid expansion of the digital economy has resulted in a substantial increase in data volume generated by various applications and systems. This surge presents significant challenges in ensuring secure and trustworthy data sharing across institutions and systems. In the realm of IoT systems and industrial production sectors, sensor networks produce vast amounts of data crucial for tasks such as

security analysis and data forecasting. Despite its value, ensuring the secure storage, reliability, and trustworthy sharing of this data remains a formidable challenge.

Regarding data storage, conventional distributed storage systems distribute data across multiple devices to balance the storage workload, offering high capacity, superior performance, and scalability [1]. However, these systems face challenges related to centralized management and lack of traceability, leading to difficulties in data sharing, susceptibility to tampering, and lack of auditability. Cloud storage systems offer an alternative, but concerns about data sovereignty, especially for core enterprise data, raise apprehensions and may need to meet stringent data confidentiality requirements.

Regarding data sharing, blockchain technology emerges as a decentralized, immutable, traceable, and collectively maintained distributed database [2]. Blockchain serves as a storage system, ensuring the traceability of textual data, provides verifiable storage, and facilitates data sharing. Nonetheless, blockchain has limitations in capacity, throughput, and latency. As of October 2023, Bitcoin's Blockchain Size has exceeded 500 GB [3], and Ethereum's blockchain size has surpassed 1300 GB [4].

Traditional data validation methods, such as hash functions and Merkle trees [5], can confirm data correctness but cannot rectify or recover erroneous data. Array codes [6], encode original data blocks into specific check codes through simple XOR and cyclic shift operations. By employing these check codes and partial original data, errors in data can be identified and corrected through these operations.

To address these challenges, this study investigates the reliable data storage and trustworthy data sharing framework based on blockchain and distributed storage technologies. The contributions of this paper are as follows:

1. A collaborative storage scheme for on-chain and off-chain data is designed, achieving high-capacity, high-performance, traceable, and verifiable data storage and access. Carefully crafting data storage and data sharing processes, and designing a unified access interface.

2. A smart contract and its invocation interface based on Hyperledger Fabric are designed, enabling traceability, verification and recovery of file information, thereby addressing the need for traceable storage.

3. Utilizing the principles and features of IPFS Cluster, an easily scalable off-chain system and interface are designed, ensuring reliable storage of massive raw data.

4. For the issue of data verification and recovery, by integrating Array code technology, the coding and storage processes are designed, realizing collaborative verification and recovery functions for on-chain and off-chain data, further enhancing the trustworthiness of shared data.

The subsequent sections of this paper are organized as follows: Section 2 introduces related work. Section 3 presents related technologies. Section 4 outlines the system design. Section 5 describes the methods for trustworthy data sharing. Section 6 conducts experimental evaluations. Section 7 concludes the study and outlines future work.

2 Related Works

2.1 Blockchain-Based Data Sharing

Blockchain technology has garnered significant attention in recent years, finding diverse applications across various fields. Researchers have explored methods to integrate blockchain, enhancing the credibility of data sharing processes [7].

In medical data sharing, Praveena Anjelin et al. pioneered a blockchain-based model [2]. Their approach involved categorizing medical institutions and implementing an enhanced consensus mechanism, ensuring secure and swift data sharing. Another noteworthy system, MedRec, introduced by Azaria et al. [8], operating on the Bitcoin network, employed proxy re-encryption technology to ensure secure data access. However, these methods were based on public chains, facing challenges such as low capacity, low throughput, and high latency. Solutions based on consortium chains are actively being pursued to address these limitations.

Zheng et al. [9] addressed the problem of extensive credit investigation data and privacy protection. The model adopts a consensus mechanism to solve the problem of large credit investigation data and privacy protection of credit investigation data and realizes access control and management of the shared data chain. In their model, data are stored on the cloud server and shared with the proxy reencryption method. To address credit data queries, Liu et al. [10] proposed a dual-blockchain integrated solution for credit data storage and query. Their approach employed two chains: one for real-time credit data collection results and another for credit data query records. This dual-chain strategy effectively mitigated concerns about excessive centralization associated with credit data collection.

Zhu et al. [11] proposed a novel system architecture that leverages blockchain technology to enhance the traceability of original achievements. Zhu et al. [12] investigated how to use blockchain technology to enhance the traceability of infectious diseases. But the authors do not discuss the scalability issues that might arise when the system is deployed in a real-world scenario with a large number of users.

Qiu and his team designed an Edge-IoT framework based on blockchain called "EdgeChain" [13]. The core idea is to integrate a permissioned blockchain system into the edge cloud resource pool and each IoT device's account and resource usage, thereby linking the behavior of IoT devices. Despite the security advantages of blockchain, the cost of integrating EdgeChain into specific IoT systems is a factor that urgently needs to be considered.

Ali et al. [14] proposed a framework based on blockchain and multiple certificate authorization. The policies and methods proposed overcome the disadvantages and security vulnerabilities faced by single certificate authorization. However, this paper only designs a detailed process for access control, does not provide a large-capacity storage infrastructure for electronic medical records, and cannot achieve large-capacity data storage.

2.2 On-Chain and Off-Chain Collaboration

On-chain and Off-chain Collaboration [15] involves storing raw data off-chain while maintaining hash values and index information on-chain, facilitating collaborative data storage and access. In a broader context, it also includes aspects such as on-chain performance optimization, off-chain storage scalability research, and collaborative computing.

To tackle data inconsistencies between on-chain and off-chain data in blockchain applications, Zhang et al. [16] proposed ĐArcher. This method detected synchronization flaws between on-chain and off-chain data, enhancing DApps' security and reliability. However, it did not optimize the reduction of test cases and execution time or handle changes in on-chain and off-chain interaction patterns.

Addressing secure search concerns in cloud storage, Li et al. [17] introduced a blockchain-based searchable symmetric encryption scheme (SSE). Data indexes and access control policies were stored

on-chain, while the content was encrypted and stored in off-chain nodes. This approach effectively achieved data privacy protection and integrity verification. Hu et al. [18] proposed an SSE solution based on Ethereum to reduce unauthorized access by malicious nodes. Chen et al. [19] introduced a blockchain-based searchable encrypted cloud solution, designing an index structure based on complex logical expressions to enhance search efficiency and scalability.

However, these solutions were based on public chains, needing admission mechanisms for participating nodes, compromising the security of on-chain data storage.

Sun et al. [20] proposed a solution using blockchain to store indexes and access control policies for electronic medical records. The original data of electronic medical records stored in the InterPlanetary File System (IPFS) was encrypted. Hao et al. [21] introduced a solution enabling dynamic authorization and revocation of data through distributed hash tables, encryption algorithms, Merkle trees, and proxy re-encryption technology.

Addressing privacy issues in cloud environments, some scholars have conducted research. Zhu et al. [22,23] investigated the impact of live streaming on online purchase intention. Kaur et al. [24] introduced a platform combining blockchain and cloud storage for managing electronic medical records in a cloud environment. But their work did not showcase the implementation details and technical challenges of blockchain technology in a cloud environment. Zhang et al. [25,26] proposed a privacy-preserving online multi-task assignment scheme to deal with privacy issues in cloud-based neural networks. Zhang et al. [27] investigated how to enforce readability and editability governance in blockchain databases using cryptographic methods. Wang et al. [28–30] provided an effective method for managing dynamic VM placement to alleviate resource contention between VMs in the data center. However, the experimental results of these paper are based on the CloudSim simulator, which may not fully reflect the situation in a real environment.

However, some scholars relied on code simulations of blockchain systems, while others implemented solutions on public chains, leaving feasibility and performance open for discussion. This study employed the mature third-generation open-source consortium blockchain platform Hyperledger Fabric for on-chain systems, optimizing performance and scalability. For off-chain systems, this study deployed a private network of multiple IPFS nodes and innovatively introduced IPFS Cluster for cluster management, optimizing cluster expansion and data arrangement. Additionally, this study proposed on-chain and off-chain collaborative verification based on erasure codes, further enhancing data integrity verification and protection.

3 Related Technologies

In this section, we present the related technologies underpinning our collaborative storage scheme. The on-chain system comprises blockchain technology and smart contracts, while the off-chain system component is a distributed storage system. Furthermore, the Array code theory provides the basis for data reliability assurance within our scheme.

3.1 Blockchain and Smart Contracts

Blockchain, a decentralized, immutable, and traceable distributed database [1], interconnects transaction records in blocks, forming an ever-expanding chain. Its core principles encompass decentralization, distributed consensus, encryption algorithms, and block linking, ensuring secure data storage, immutability, and trustworthiness.

Decentralization in blockchain disperses data and power across numerous network nodes, mitigating single points of failure and bolstering system reliability [31]. Encryption algorithms and consensus mechanisms fortify data security; each block contains the previous block's hash value, thwarting tampering and preserving immutability.

Blockchain is categorically divided into public chains, private chains, and consortium chains [32]. Public chains, open to all, epitomize decentralization and transparency. Private chains restrict participation, tailored for specific entities and internal use. Consortium chains, overseen by multiple entities, feature nuanced permissions and consensus mechanisms, vital in collaborative business contexts. Prominent consortium chains encompass Hyperledger Fabric [33] and FISCO BCOS [34].

Smart contracts, autonomously executing on the blockchain, operate on code and conditional rules [35]. They trigger transactions or business logic upon meeting specific criteria, ensuring dependable, intermediary-free transactions and cooperative relationships across diverse applications.

3.2 Distributed Storage Systems and Scalability

Distributed storage systems distribute data across multiple nodes, delivering high-capacity, high-performance, and exceedingly reliable storage solutions [36]. This system segments data into blocks or objects, storing them across various nodes. Through distributed storage and parallel processing, these systems optimize capacity, throughput, and availability.

These systems consist of interconnected nodes, each equipped with storage media, processors, and network interfaces. The architecture integrates metadata servers [37], data distribution strategies [38], data consistency protocols, and fault tolerance mechanisms [39], guaranteeing data reliability and consistency.

Prominent distributed storage systems include GFS [36], Ceph [40], HDFS [41], Lustre [42], and IPFS [43]. GFS, created by Google, remains proprietary. HDFS, a Hadoop subproject, excels in large-scale data but may not suit extensive multimodal data storage. Lustre, an open-source, high-performance parallel file system, finds widespread use in supercomputing. Ceph, designed for high performance, reliability, and scalability, provides unified distributed file services. IPFS, a decentralized peer-to-peer file system, utilizes content addressing and peer-to-peer technology, enabling decentralized data storage and sharing.

Distributed storage systems exhibit remarkable scalability and performance [44]. Scalability permits seamless addition of storage nodes and processing capabilities to accommodate burgeoning storage needs. These systems employ data sharding, caching, parallel processing, and load balancing to enhance performance, bolstering response speed and throughput.

3.3 Array Codes

Array codes [6] are a special class of erasure codes, which involve only simple XOR and cyclic shift operations in both the encoding and decoding procedures, thus are much more efficient than the well-known Reed-Solomon codes in terms of computational complexity. These codes have been widely used in RAID architectures and distributed storage systems in recent years. Our research group has obtained several important results in array codes, including lowest density array codes [45–47], highly fault-tolerant array codes [48], efficient encoding and decoding algorithms for array codes [49,50]. It is worth mentioning that, the RΛ-Code [47] and the generalized [48] remains the most efficient lowest density array codes and highly fault-tolerant array codes, respectively. Therefore, these codes will be used in our designed collaborative storage system.

4 System Design

In the following sections, we present our overall design for a collaborative storage system that integrates both on-chain and off-chain technologies. We then provide the data flow among system components and roles, followed by a detailed discussion on the design of our smart contracts. Lastly, we explore the process of cluster expansion and dynamic node management.

4.1 System Architecture

The on-chain and off-chain collaborative storage system combines blockchain and distributed storage technology to achieve secure, trustworthy, and efficient file storage and access functionalities. The on-chain system utilizes blockchain technology to store file metadata and verification data, ensuring the immutability and transparency of the stored information. Meanwhile, the off-chain system leverages distributed storage technology, providing highly accessible file storage and retrieval functionalities. Users benefit from a unified access interface, simplifying operations like file upload, download, and queries. Additionally, the system incorporates database components to facilitate efficient index creation and management. This collaborative approach ensures data security, scalability, and flexibility, thereby delivering adequate file storage and access services to meet the demands of a growing user base.

Fig. 1 presents the hierarchical structure of our collaborative storage system, which is divided into three layers. The Application Layer integrates various components into the Collaborative Storage Server, including Unified Access, File Verification, ECC Encoding/Decoding, Contract Invocation, and Data Storage/Access Services Provider. The Tracing Layer employs blockchain and our custom smart contracts for managing metadata, hashes, RΛ-Code check blocks, and more, thereby achieving decentralization, traceability, access control, and trusted traceability. The Storage Layer utilizes a distributed storage cluster for data storage, load balancing, decentralization, fault tolerance, resilience, redundancy, and data storage/access.

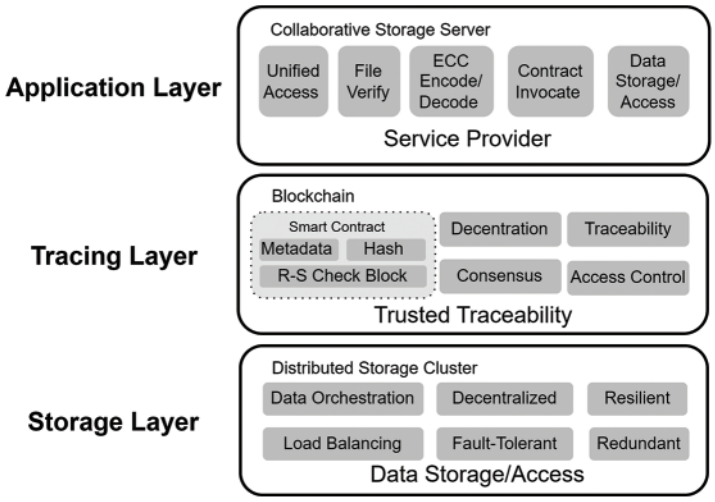

Figure 1: Hierarchical architecture and component functions of the collaborative storage system

The diagram depicted in Fig. 2 provides a comprehensive overview of the system's architecture, which includes Collaborative Storage Servers, a blockchain network, and a distributed storage cluster. Each component has distinct functions, elaborated below:

Distributed Storage Cluster: Constructed using IPFS Cluster, this cluster comprises multiple storage nodes, each responsible for storing file data. Within the off-chain system, there are storage nodes and monitoring nodes. These storage nodes, driven by IPFS, facilitate fundamental file storage and access capabilities. IPFS Cluster supervises the replication and sharing of data across numerous IPFS nodes. Each node runs the IPFS node daemon and IPFS Cluster Service daemon, establishing connections with other nodes to create a decentralized network where all nodes are equal peers. This off-chain system guarantees high availability, fault tolerance, and scalability, ensuring secure storage and rapid file access within a distributed environment.

Blockchain Network: Built on the Hyperledger Fabric blockchain platform, this network primarily manages file metadata and verification data. Leveraging the decentralized and tamper-proof features of blockchain technology it ensures the integrity and traceability of files. The on-chain system achieves metadata storage and verification data preservation through smart contracts, aiming to provide secure and reliable file information traceability and data recovery functionalities.

Collaborative Storage Servers: These servers seamlessly integrate on-chain and off-chain data interfaces, featuring a unified access interface for user-friendly services. Users can store data in the collaborative storage system by uploading files through the data storage interface. The system automatically generates multiple copies of the original data in the off-chain system, extracts metadata and verification data, and stores them in the on-chain system, ensuring the authenticity and reliability of the data. Additionally, users can download file content from the off-chain system through the data retrieval interface for local access and usage. Moreover, users can efficiently query file information from either the on-chain or off-chain sources using the query interface.

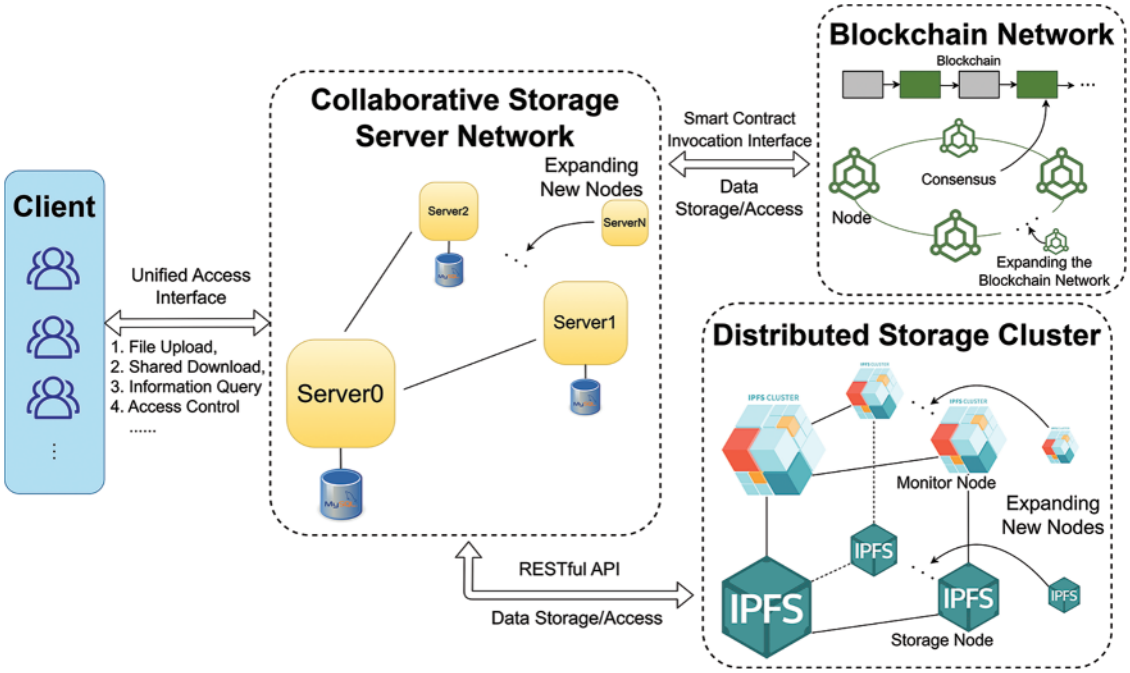

Figure 2: On-chain and off-chain collaborative storage system architecture

4.2 Data Flow among System Components and Roles

As Fig. 3 shows, during the data upload process, the file owner first registers and establishes an account within the system. Subsequently, the file is uploaded through the designated account. In this process, ServerN extracts crucial information such as metadata, hash values, RΛ-Code check blocks, and share tags. These details are securely stored in the blockchain network, ensuring the immutability and transparency of the data. Simultaneously, ServerN stores the original file in the distributed storage cluster to guarantee high availability and rapid access. This procedure ensures the secure and reliable preservation of uploaded data within the system.

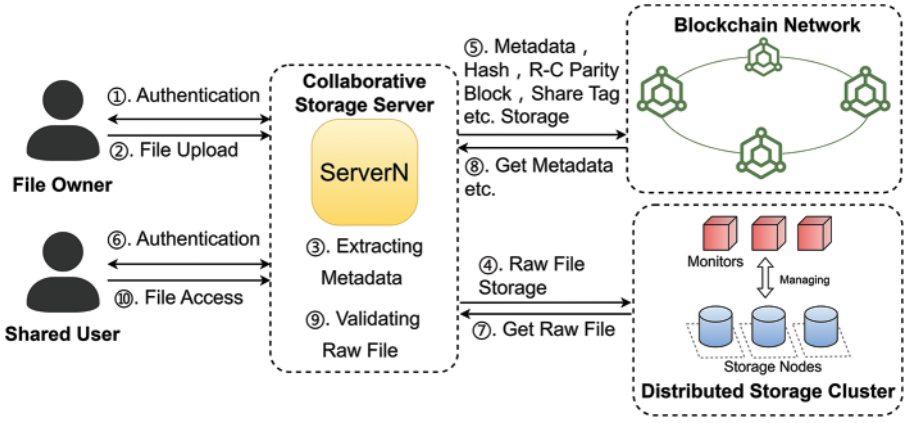

Figure 3: Comprehensive data flow for data upload and download processes

In the data download process, shared users undergo registration and account creation to gain system access. When users require specific file details, they query the system for file information, including metadata and other relevant data stored on the blockchain network. Upon requesting file access, ServerN retrieves the original file from the distributed storage cluster. Subsequently, ServerN utilizes the pertinent data from the blockchain to validate the integrity of the original file and delivers it to the user, allowing local access and utilization. This process ensures that users efficiently obtain file information while guaranteeing the integrity and reliability of the downloaded files. Through these operations, the system plays a pivotal role in safeguarding data security and user convenience during both the data upload and download processes.

4.3 Smart Contract Design

Smart contracts, known as chaincodes in Hyperledger Fabric, represent operational logic codes within the blockchain network. They handle data management tasks on the blockchain, defining functions and rules for operations such as data creation, reading, updating, and deletion. These contracts are executed under specific circumstances: **1. Transaction Submission:** Smart contracts validate and execute transactions when participants submit them to the blockchain network, adhering to predefined content and rules. **2. Query Requests:** Participants can request specific data through queries, and smart contracts provide relevant results based on the query's content.

The design standards for these contracts are known as the chaincode specification, outlining the interface and behavior of the code. This specification covers transaction proposal handling, state management, event triggering, and more. Detailed chaincode specifications are in official documentation, providing comprehensive explanations and example code snippets.

The smart contract developed in this study manages diverse file-related information on the blockchain, including the Unified Identifier (CID), file metadata, hash data, parity-check blocks, and permission details. Through this contract, functionalities like file information management, integrity verification, data recovery, and access control can be implemented on-chain.

As depicted in Fig. 4, the smart contract utilizes specific data structures and functionalities to enable file storage, access control, and data integrity verification. This contract processes logic, including file information management, integrity verification, recovery of lost data, metadata updates, and uploader's permission validation.

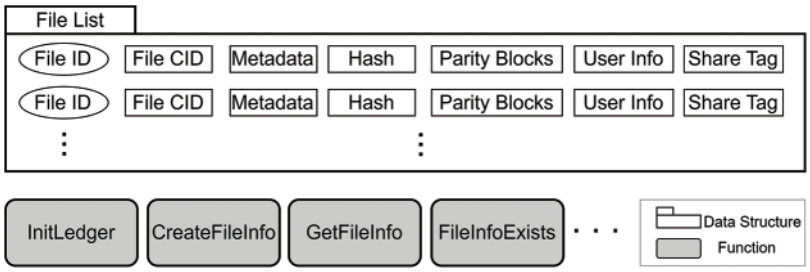

Figure 4: Smart contract architecture

The meanings and generation methods of some fields in the data structure of the smart contract are as follows: File ID: A unique identifier distinguishing different files within the system. File CID: A unique content identifier generated using encryption algorithms, representing the file's unique hash value for retrieval from off-chain systems. Metadata: Descriptive file information, including name, size, creation time, permissions, and directory status. Hash: The file's hash value used for integrity and consistency checks. Parity Blocks: Redundant blocks generated using RΛ-Code, aiding in data recovery for lost or corrupted files. User Info: User-related details associated with the file, such as the owner and shared users. Share Tag: A permission indicator determining whether the file can be shared with other users.

The smart contract methods, although not limited to those depicted, include: InitLedger: Initializes the ledger during contract deployment, setting the initial contract state, including data structures and parameters. CreateFileInfo: Creates file information when a user uploads a new file, storing metadata, hash values, and other details in the smart contract. GetFileInfo: Retrieves file details using the file's identifier (e.g., File ID) as a parameter, providing information like name, size, and upload time. FileInfoExists: Validates the existence of specific file information based on the file's identifier, determining whether the file has been recorded.

4.4 Cluster Expansion and Dynamic Node Management

4.4.1 On-Chain System Expansion

Dynamic node management within a blockchain network is implemented by the blockchain, adhering to the best practices and network security standards of Hyperledger Fabric to ensure stability, security, and scalability. This process generally involves:

Designing Organizational Structure: Defining the structure of organizations, nodes, identities, channels, and other elements, and determining roles and permissions for each organization in the network, including configurations for new nodes.

Setting up Certificate Authority (CA) Services: Utilizing Fabric's CA services to generate identity certificates and keys for organizations and nodes. New nodes require appropriate identity certificates to join the network.

Configuring Channels and Smart Contracts: Defining channels, configuring smart contracts, and ensuring newly added nodes have the necessary permissions to join channels and participate in smart contract transactions.

Updating Configuration Blocks: When new nodes join, a configuration block containing the current network configuration information must be generated. New nodes require this block to understand the network's initial state.

Joining Channels: New nodes must send requests to join channels, which other nodes must validate before allowing the new node to join.

Chaincode Installation and Instantiation: If new nodes need to execute specific chaincodes, these chaincodes must be installed on the new nodes and instantiated on the channel.

4.4.2 Off-Chain System Expansion

Dynamic node management is fundamental in distributed storage systems. In the IPFS Cluster, enabling dynamic management of new nodes involves configuring settings in the IPFS Cluster and IPFS Swarm configuration files. Alternatively, APIs can be utilized to manage the dynamic addition and removal of nodes, which generally include:

Initializing IPFS Cluster: Before launching IPFS Cluster, cluster configurations must be initialized. This includes defining the cluster's identity (Cluster ID), configuring peering settings (API endpoints, keys, and other elements), and specifying data storage locations.

Configuring IPFS Swarm: Ensuring correct configurations for IPFS Swarm (the underlying communication layer of the IPFS network). Parameters such as listening addresses, ports, and encryption methods must be set.

Node Identity Certificates: Ensuring that new nodes possess the correct identity certificates. TLS certificates are necessary to secure communication between nodes.

Dynamic Node Addition: When a new node joins the cluster, it must connect to the cluster's API endpoint and undergo appropriate authentication. IPFS Cluster provides APIs for node management, allowing new nodes to request cluster entry.

Configuring Discovery: Utilizing discovery services (such as mDNS, Bootstrap nodes) to enable other nodes to discover new nodes and integrate them into the IPFS network.

Dynamic Node Removal: IPFS Cluster typically automatically detects inactive nodes. Suppose a node remains offline for an extended period. In that case, the cluster marks it as unavailable, automatically adapting by migrating storage and load requirements to other nodes to maintain overall system stability and performance.

4.4.3 Collaborative Storage Server Expansion and Dynamic Node Management

The objective of the dynamic node management protocol is to facilitate the dynamic addition and removal of nodes in distributed systems, ensuring cluster vitality and stability. This protocol enables new nodes to join the distributed system at runtime while allowing nodes to securely exit without affecting the entire system. Through periodic heartbeat messages, the protocol ensures the activity and connection status of every node, preventing issues such as node disconnection due to faults or other

reasons. When a node exits due to failure or other reasons, the cluster adapts automatically, migrating storage and load requirements to other nodes, ensuring the overall stability and performance of the system.

Firstly, a unique identifier and public-private key pair are generated for each node in the cluster. Subsequently, each node regularly sends heartbeat messages to indicate its activity status. These messages should include the sender's identifier, status information, timestamp, and other elements. A heartbeat interval is defined. Each node maintains a node status table, recording activity status, the last heartbeat time, and other information about other nodes. If a node does not receive a heartbeat message from another node for a prolonged period, it can mark that node as unavailable. When a new node wishes to join the cluster, it can send a join request to any node in the cluster. The receiving node can verify the new node's identity, add it to the node list, and inform other nodes. Upon receiving the notification, the new node can send heartbeat messages to other nodes. When a node intends to exit the cluster, it can send an exit notification to other nodes in the cluster. The receiving nodes should remove the exiting node from the node list and notify other nodes. The removed node stops sending heartbeat messages. If a node does not receive heartbeat messages from another node for an extended period, it marks the node as unavailable and initiates node replacement.

5 Methods for Trusted Data Sharing

5.1 Data Storage Process

The data storage process aims to receive files uploaded by users, store them in the server cache, save the original files in the IPFS Cluster, store metadata and verification data in the Fabric network, and index information in the database. As Fig. 5 shows, here is a detailed overview of the execution flow:

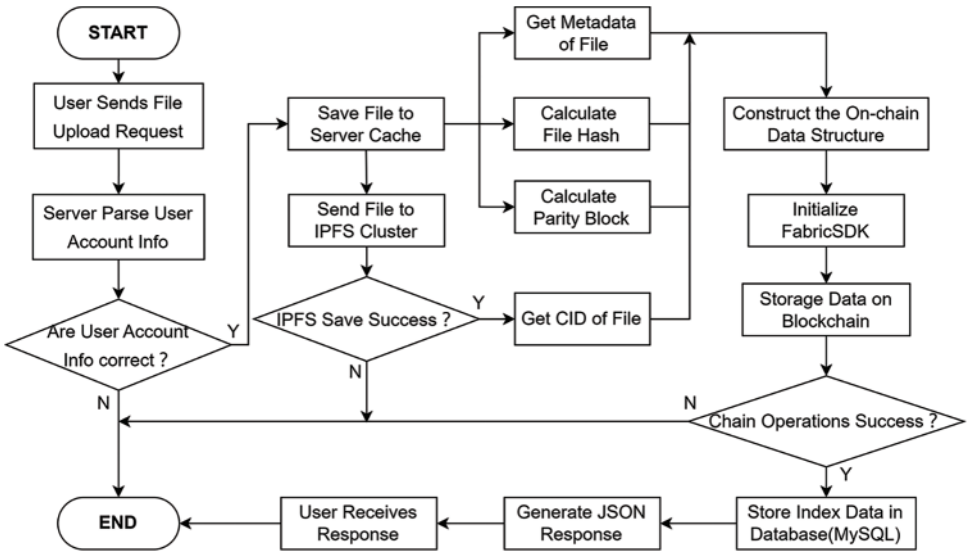

Figure 5: Data storage flowchart

Upon receiving a file upload request, the server parses the user's account information from their previous login, capturing details like username and user ID for on-chain ownership identification. Any errors trigger an error message response. The server records the file reception start time, creates a user

directory, defines the storage path, and saves the uploaded file in the server cache. It then calculates reception end time and latency consumption.

Simultaneously, a goroutine is initiated to save the file to the IPFS system. A channel is established to receive IPFS file writing results, and the start time of this IPFS data write operation is noted. IPFS File Saving Operation: The file is sent to the IPFS system for storage, and the result is communicated through the channel. The system waits for IPFS upload results and the completion of on-chain operations. In case of IPFS failure, an error message is returned. If successful, a JSON response is generated, including reception duration, IPFS writing time, and output information, and sent back as a confirmation.

Concurrent tasks handle file metadata retrieval, hash calculation, and parity-check block computation. Fabric SDK facilitates on-chain data transactions. The operation duration and potential errors are returned.

Extracting parity-check blocks involves reading the specified input file, encoding it using RΛ-Code encoder, and writing the last parity-check block to the output file. Here are the steps: Firstly, the designated input file is opened. If unsuccessful, an error message is generated, and an error is returned. Details about the input file, including its size, are retrieved. Block size (256 KB), the number of data blocks (dataShards), and the total number of data and parity-check blocks (totalShards) are calculated. The RΛ-Code encoder is initialized, and a two-dimensional byte slice is created to store data and parity-check blocks. The file is read in a loop for encoding, and a data block slice is created to store the read data. If a non-EOF read error occurs, an error message is generated, and an error is returned. If the read data is less than a block's size, zero padding is applied. The data and parity-check blocks are passed to the encoder for encoding. If an encoding error occurs, an error is returned. An output file is created to store the encoded data, and the last parity-check block is written to this file. If writing to the output file fails, an error message is generated, and an error is returned. Finally, nil is returned, indicating no errors occurred.

A goroutine is initiated to execute on-chain operations. A channel is established to receive results from both uploads and on-chain operations. The start time of these operations is recorded. Once executed, the results are sent to the channel. If both upload and on-chain operations fail, an error message is returned. If successful, a JSON response confirming on-chain data storage, including upload and on-chain operation duration, is generated and returned. Finally, file index information is stored in the database.

5.2 Trusted Data Sharing Process

The Trusted Data Sharing Process serves to facilitate data download and recovery. It employs multiple threads to acquire verification data and original files from Fabric and IPFS. Subsequently, leveraging the obtained verification data, it verifies the integrity of the files and conducts necessary recovery operations if required. Finally, the process furnishes the client with download links, verification results, and metadata in JSON format. This information includes relevant status updates, download links, and detailed time consumption statistics.

As Fig. 6 shows, initially, the process entails parsing the request parameters and validating user permissions. The system establishes a dedicated channel (ch) to handle data flow efficiently, acting as a conduit for receiving hash data and verification information from Fabric, with the process's start time being meticulously recorded.

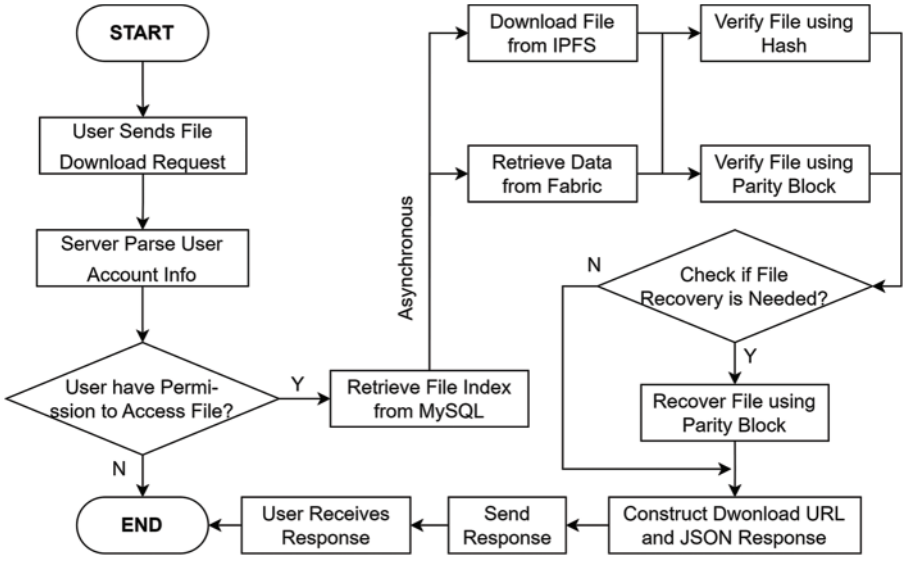

Figure 6: Data download and verification flowchart

A dedicated thread is initiated to retrieve hash data and verification details from Fabric. This operation is facilitated through the initialization of the Fabric SDK, establishing a crucial link with the Fabric network. Comprehensive file details are then extracted, specifically reading detailed information of the designated CID file. The acquired data is promptly dispatched to channel ch for further processing.

Simultaneously, another thread is launched to download files from IPFS, employing the IPFS client. During this operation, the file's destination path is defined, and the precise start time for the IPFS download process is noted. In the event of any download errors, these issues are meticulously logged and transmitted to the error channel. The overall time taken for the IPFS download process is accurately calculated.

Verification data procured from channel ch is stored in a designated file. Subsequently, the process systematically verifies the file's integrity. If no recovery is necessitated, the process accurately calculates the duration required for file recovery. In such cases, a null value is dispatched to the error channel. Conversely, if file recovery is imperative, the process seamlessly executes recovery operations utilizing the RΛ-Code.

The integrity of the file is scrutinized through erasure coding mechanisms. This approach assesses the file's completeness and triggers data recovery as needed. Fundamental steps include opening the input file, retrieving essential file information such as size, and calculating the number of data blocks (dataShards) and total blocks (totalShards). Initialization of the RΛ-Code encoder follows, wherein appropriate settings for data blocks and parity-check blocks are configured. A two-dimensional byte slice (data) is generated to meticulously store original data blocks and parity-check blocks. File content is read in manageable chunks, with the final parity-check block extracted from the parity file and stored appropriately. Erasure coding is then meticulously executed to determine the necessity for data recovery. In cases of verification failure, meticulous data recovery processes are activated. An output file is created to store the recovered data securely. Conversely, if verification attests to the file's integrity, signifying that no recovery is needed, pertinent information is relayed accordingly.

Lastly, a comprehensive JSON response is crafted, encapsulating diverse status updates, download links, and detailed time-related statistics. This JSON response undergoes encoding into a string format, accompanied by configuration of response headers. The finalized response is then disseminated.

6 Experimental Evaluation

In this section, we explore the experimental setup and performance evaluation of our collaborative storage system. We compare the capacity of our system under different erasure coding parameters. We conduct system reliability tests to assess the system's fault tolerance and recovery capabilities under various abnormal conditions. Finally, we evaluate the system's latency and throughput using diverse open-source datasets.

6.1 Experimental Environment and Deployment Information

This section provides a comprehensive overview of the deployment process for the experimental hardware and software setups, detailing the configurations of the Hyperledger Fabric system and the IPFS Cluster cluster. All components of our collaborative storage system were distributed across three servers, as detailed in Tables 1 and 2.

Table 1: Hardware environment of three servers

Item	Description
CPU	Intel (R) Xeon (R) Silver 4216 CPU @ 2.10 GHz
Memory	Samsung 2400 MHz DDR4 96 GB
SSD	WD BLACK SN770 500 GB

Table 2: Software environment of all servers

Item	Description
OS	Ubuntu server 20.04 LTS
Golang	Version 1.20.3
MySQL	Version 8.0.33
Kubo (IPFS)	Version 0.19.1
Ipfs-cluster-service	Version 1.0.6
Hyperledger-fabric	Version 2.5.0
Hyperledger-fabric-ca	Version 1.5.6

In this deployment, IPFS components were distributed across the three servers, and all IPFS Cluster components were co-located. Hyperledger Fabric was containerized, employing three containers on the first server to simulate three nodes. The Hyperledger Fabric CLI and chaincode were containerized and deployed on the first server as well. Hyperledger Explorer was deployed on the first server, with port forwarding configured on the host machine, facilitating seamless access to Hyperledger Explorer through the browser. All other necessary software packages were systematically deployed across these three servers.

6.2 Capacity Comparison

In the collaborative storage system, raw data is stored off-chain, while metadata and erasure coding check blocks are stored on-chain. The capacity improvement of the collaborative storage system under different sizes of erasure coding check blocks depends on the specific parameters of erasure coding and the on-chain storage capacity. Fig. 7 illustrates the storage capacity of the collaborative storage system relative to the blockchain system under different ratios of erasure coding check blocks, with its storage capacity being 4 to 10 times that of the blockchain system. It is evident that the collaborative storage system significantly reduces on-chain storage requirements, enhancing storage efficiency while ensuring data reliability and security.

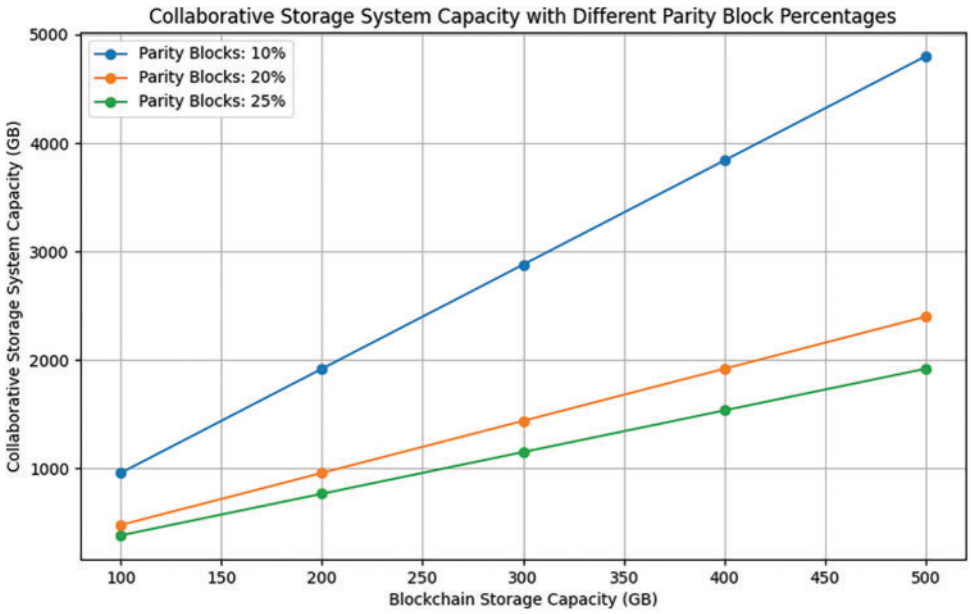

Figure 7: Storage capacity of collaborative storage system relative to blockchain system under different erasure coding check block ratios

6.3 System Reliability Testing

System reliability testing is a crucial step in ensuring the system operates smoothly under various abnormal conditions. In the blockchain system, all data is fully replicated, with each node possessing all data. In our implemented system, the redundancy parameter for distributed storage is set to three copies. In the reliability tests, various potential crash scenarios were tested to verify the system's fault tolerance and recovery capabilities.

During distributed storage node crash tests, two distributed storage nodes were manually shut down to simulate node failures. We added an additional node and the system's data redistribution and successful data pinning to other nodes were observed. Additionally, user access to data was tested to ensure normal functionality. When a node fails, the cluster automatically re-pins the file to a new node, ensuring data redundancy. This ensures that the off-chain system continues to operate normally and provides the required file services. Fig. 8 displays the automatic re-pinning log information for a data block in case of an abnormality.

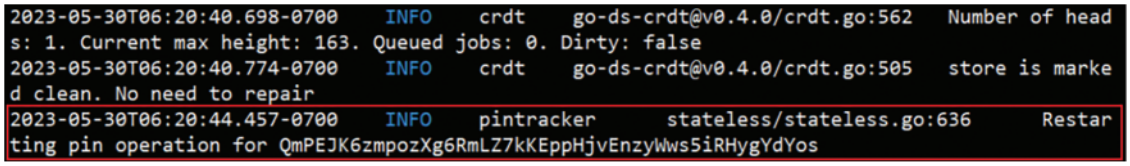

Figure 8: Log information of IPFS cluster automatically pinning data blocks

In tests simulating blockchain node failures, one container was manually closed to simulate a blockchain network failure. The response of the blockchain client was observed, checking if blockchain data could be obtained through other nodes or backup nodes. The results demonstrated that the system could continue to obtain blockchain data through other nodes, ensuring the integrity and availability of blockchain data.

During collaborative storage server crash tests, services on one node were manually shut down. The system load seamlessly switched to other nodes, ensuring the ability to continuously provide file upload and sharing services.

6.4 Latency and Throughput Testing

For performance testing, diverse open-source datasets were used for file storage and sharing tests. File sizes ranged within the expected processing scope of our system, gradually increasing, with some files surpassing 500 MB for comprehensive performance evaluation. Test scripts were utilized to construct file upload requests and record results. These automated tests saved results persistently, capturing each upload's start and end times and facilitating upload latency calculation.

Fig. 9 illustrates the latency to file size concerning (a) file upload, (b) erasure coding, (c) query, (d) download. Generally, upload latency increases with file size. In Fig. 9a, "Upload Duration" signifies total upload time, "IPFS Write Time" indicates time taken to store the original file in the off-chain IPFS Cluster system, and "Chain Data Time" represents latency for data extraction and storage in the on-chain Fabric system. Notably, upload latency gradually rises with file size, exceeding 30 s for files larger than 500 MB. Large file upload latency is mainly due to the distributed storage system's file storage process; blockchain consumption is comparatively minor. Our system's performance is on par with enterprise-level storage services. On-chain data storage time ranges from 0 to 200 ms, well within acceptable limits.

In Fig. 9b, "Encode and Read Parity Time" illustrates latency to file size for encoding and reading parity check blocks using RΛ-Code relative. Erasure coding time is significantly smaller than the time taken to store files in the distributed storage system. In Fig. 9c, "Ipfs Delay" represents the time taken to query file information from off-chain IPFS, and "Fabric Delay" represents the time taken to query file information from on-chain Fabric. Query latency for file-related metadata remains within 60 ms. In Fig. 9d, "Receive Time" indicates the time taken to obtain files from the server. Overall, the download time is considerably shorter than the upload time and falls within acceptable limits.

Fig. 10 demonstrates throughput to file size for (a) file upload, (b) download. Clearly, under gigabit network bandwidth, our collaborative storage system achieves a maximum upload throughput of 38 MB/s, averaging 20 MB/s. During file downloads, maximum throughput reaches 50 MB/s, averaging 30 MB/s. Compared to using the blockchain alone, the throughput of the collaborative storage system has significantly improved. When processing test files ranging from 100 to 500 MB, its throughput surpasses the on-chain system by a factor of 4 to 18.

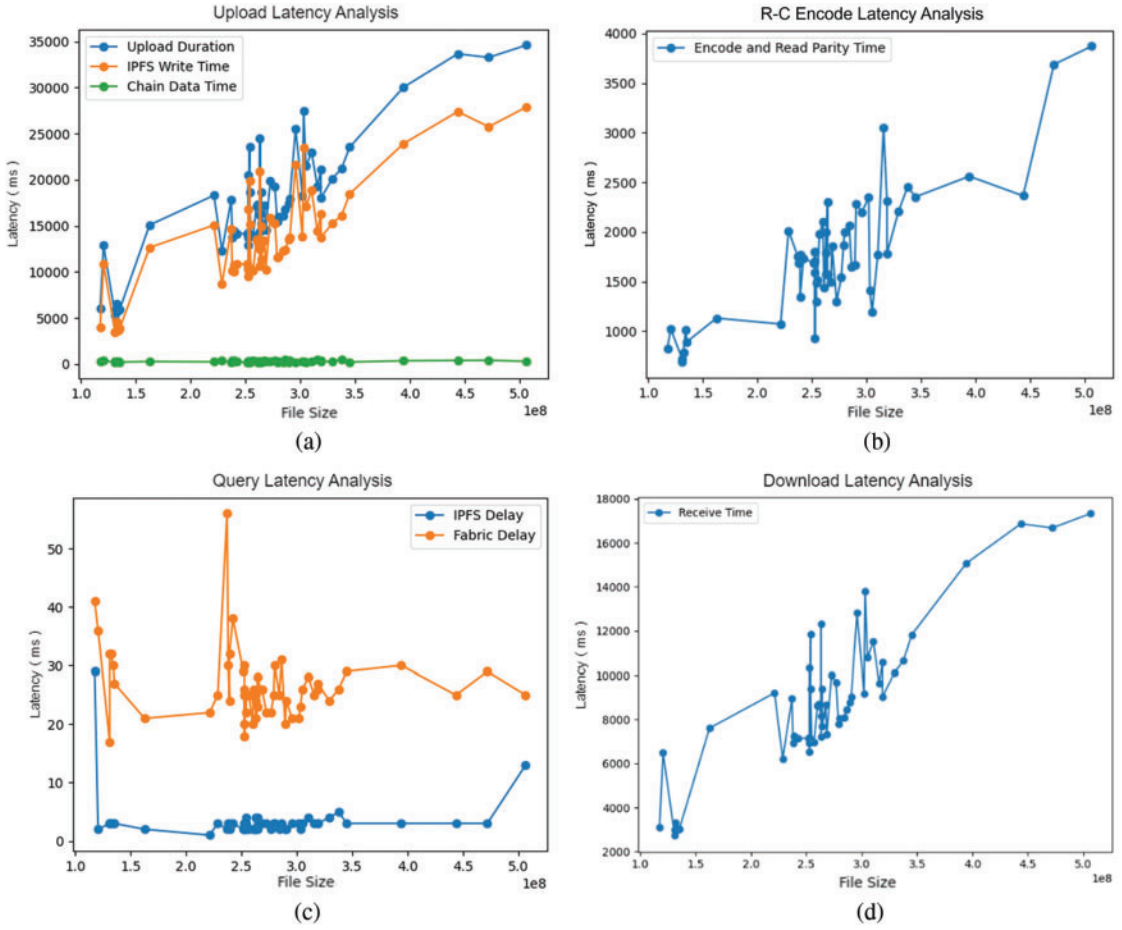

Figure 9: Performance by file size for (a) File upload, (b) RΛ-Code encoding, (c) Query, (d) File download

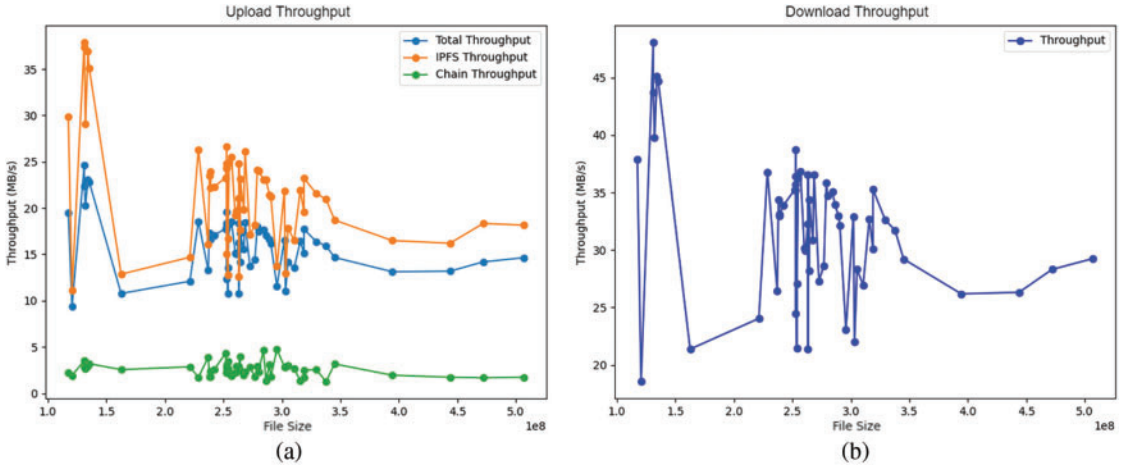

Figure 10: Performance by file size for (a) File upload, (b) File download

7 Conclusion and Future Work

This study explores the reliable storage and trustworthy sharing of data in Internet of Things (IoT) systems. To address the limitations of traditional systems and meet the increasing demand for data security, we propose an innovative system architecture integrating on-chain and off-chain collaboration. This architecture merges blockchain and distributed storage technologies, achieving high-capacity, high-performance, traceable, and verifiable data storage and access functions. This approach supports the optimization and decision-making processes within IoT systems.

In the on-chain system, we utilize Hyperledger Fabric, incorporating smart contracts and smart contract invocation interfaces. This design facilitates the management of metadata, validation data, and permission information for raw data. Simultaneously, in the off-chain system, we employ the IPFS Cluster, designing interfaces that ensure reliable storage and efficient access to large-scale files. We develop a collaborative storage server that integrates both on-chain and off-chain operation interfaces, enabling extensive data collaboration and providing users with a user-friendly interaction experience. Simultaneously, by formulating sensible data sharing mechanisms and privacy protection strategies, secure data sharing can be realized, fostering data reuse. By integrating RΛ-Code technology, the design of the coding and storage processes enables collaborative verification and recovery functions for both on-chain and off-chain data.

Empirical findings substantiate that our collaborative storage system significantly enhances storage capacity compared to the standalone use of blockchain. Regarding system fault tolerance, it can withstand the failure of all other copies of a specific piece of data, ensuring uninterrupted service provision. Furthermore, the system exhibits high scalability and incorporates an automatic disaster recovery mechanism. Concerning data integrity verification, our methodology employs a triple-redundant protection mechanism encompassing redundant storage, hash verification, and erasure code verification, thereby ensuring secure data storage and trustworthy shared data. Regarding system performance, extensive stress tests conducted on open-source datasets reveal that its throughput surpasses the on-chain system by a factor of 4 to 18.

This study, implemented on the third-generation blockchain platform Hyperledger Fabric, known for its superior performance and scalability. While existing research predominantly focuses on public chains, discussions related to consortium chains often neglect system scalability.

Furthermore, our study introduces the IPFS Cluster for cluster management, as previous scholars have yet to explore this avenue. Many existing IPFS-based studies restrict their focus to single-node setups, neglecting the potential of distributed clusters. Our study enhances cluster management and data orchestration, improving system performance and scalability.

An innovative aspect of our study is proposing a collaborative verification method based on erasure codes for on-chain and off-chain data. In traditional research, erasure codes are only employed for data verification in distributed storage systems. Our method significantly bolsters data reliability.

In conclusion, this paper materializes a collaborative on-chain and off-chain storage system based on blockchain and distributed storage, affirming its feasibility and effectiveness in IoT data storage and sharing. Future endeavors could focus on optimizing system performance and exploring applications in more practical scenarios. Additionally, we have observed significant fluctuations in system performance, which warrants further investigation into the causes of these fluctuations and subsequent optimization in future studies.

Acknowledgement: We are grateful to the anonymous reviewers and editors for their very constructive comments and consideration of the publication of our paper.

Funding Statement: This work is supported by the National Key Research and Development Program (No. 2022YFB2702101), Shaanxi Key Industrial Province Projects (2021ZDLGY03-02, 2021ZDLGY03-08), and the National Natural Science Foundation of China under Grants 62272394 and 92152301.

Author Contributions: Jinyang Yu designed the entire experiment, implemented the framework of the entire system, and drafted the paper. Xiao Zhang provided constructive guidance and revisions for the paper and the experiments. Yuchen Zhang assisted in completing the on-chain and off-chain system testing section. Yulong Shi and Linxuan Su assisted in the design and implementation of the collaborative storage system. Jinjiang Wang and Leijie Zeng assisted in the drafting, revising, and proofreading of the paper, as well as the creation and modification of key figures. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: Jinyang Yu, Xiao Zhang, Jinjiang Wang, Yuchen Zhang, Yulong Shi, Linxuan Su and Leijie Zeng declare no conflict of interest.

References

[1] J. Zou, A. Iyengar, and C. Jermaine, "Architecture of a distributed storage that combines file system, memory and computation in a single layer," *VLDB J.*, vol. 29, pp. 1049–1073, 2020. doi: 10.1007/s00778-020-00605-w.

[2] D. Praveena Anjelin and S. Ganesh Kumar, "Blockchain technology for data sharing in decentralized storage system," In: S. S. Dash, S. Das, B. K. Panigrahi (Eds.), *Intelligent Computing and Applications. Advances in Intelligent Systems and Computing*, Singapore: Springer, 2021, vol. 1172.

[3] "Bitcoin Blockchain Size," YCharts, 2023. [Online]. Available: https://ycharts.com/indicators/bitcoin_blockchain_size (accessed on 14/12/2023).

[4] "Ethereum Chain Full Sync Data Size," YCharts, 2023. [Online]. Available: https://ycharts.com/indicators/ethereum_chain_full_sync_data_size (accessed on 14/12/2023).

[5] H. Liu, X. Luo, H. Liu, and X. Xia, "Merkle tree: A fundamental component of blockchains," in *2021 Int. Conf. Electron. Inf. Eng. Comput. Sci. (EIECS)*, Changchun, China, 2021, pp. 556–561.

[6] M. Blaum, P. G. Farrell, Tilborg V., and C. A. Henk, "Multiple burst-correcting array codes," in *IEEE Transactions on Information Theory*, vol. 34, no. 5 pp. 1061–1066, Sept. 1998. doi: 10.1109/18.21231.

[7] T. Xiang, Y. Zhuang, W. Xiao, S. Zhou, and J. Guan, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Serv.*, vol. 14, no. 4, pp. 352–375, 2018. doi: 10.1504/IJWGS.2018.095647.

[8] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using blockchain for medical data access and permission management," in *2016 2nd Int. Conf. Open and Big Data (OBD)*, 2016, vol. 2, no. 1, pp. 25–30.

[9] K. Zheng *et al.*, "Blockchain technology for enterprise credit information sharing in supply chain finance," *J. Innov. Knowl.*, vol. 7, no. 4, pp. 100256, 2022. doi: 10.1016/j.jik.2022.100256.

[10] F. Liu, Q. Sun, and J. Li, "Fusion of double block chain credit data storage and query scheme," *Comput. Eng. Appl.*, vol. 58, no. 2, pp. 123–128, 2022. doi: 10.3778/j.issn.1002-8331.2008-0070.

[11] P. Zhu, J. Hu, X. Li, and Q. Zhu, "Using blockchain technology to enhance the traceability of original achievements," *IEEE Trans. Eng. Manage.*, vol. 70, no. 5, pp. 1693–1707, May 2023. doi: 10.1109/TEM.2021.3066090.

[12] P. Zhu, J. Hu, Y. Zhang, and X. Li, "Enhancing traceability of infectious diseases: A blockchain-based approach," *Inf. Process. Manage.*, vol. 58, no. 4, pp. 102570, 2021. doi: 10.1016/j.ipm.2021.102570.

[13] J. Pan, J. Wang, A. Hester, I. Alqerm, Y. Liu, and Y. Zhao, "EdgeChain: An edge-IoT framework and prototype based on blockchain and smart contracts," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4719–4732, 2018. doi: 10.1109/JIOT.2018.2878154.

[14] A. Ali *et al.*, "A novel secure blockchain framework for accessing electronic health records using multiple certificate authority," *Appl. Sci.*, vol. 11, no. 21, 2021. doi: 10.3390/app11219999.

[15] K. Wang, Y. Yan, S. Guo, X. Wei, and S. Shao, "On-chain and off-chain collaborative management system based on consortium blockchain," in *Proc. ICAIS*, Dublin, Ireland, 2021, pp. 172–187.

[16] W. Zhang, L. Wei, S. Li, Y. Liu, and S. Cheung, "ÐArcher: Detecting on-chain-off-chain synchronization bugs in decentralized applications," in *Proc. 29th ACM Joint Meet. Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2021, pp. 553–565.

[17] H. Li, F. Zhang, J. He, and H. Tian, "A searchable symmetric encryption scheme using blockchain," arXiv preprint arXiv:1711.01030, 2017.

[18] S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, and K. Ren, "Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization," in *IEEE INFOCOM 2018-IEEE Conf. Comput. Commun.*, Apr. 2018, pp. 792–800.

[19] L. Chen, W. K. Lee, C. C. Chang, K. K. R. Choo, and N. Zhang, "Blockchain based searchable encryption for electronic health record sharing," *Future Gener. Comput. Syst.*, vol. 95, pp. 420–429, 2019. doi: 10.1016/j.future.2019.01.018.

[20] J. Sun, X. Yao, S. Wang, and Y. Wu, "Blockchain-based secure storage and access scheme for electronic medical records in IPFS," *IEEE Access*, vol. 8, pp. 59389–59401, 2020. doi: 10.1109/ACCESS.2020.2982964.

[21] J. Hao, C. Huang, W. Tang, Y. Zhang, and S. Yuan, "Smart contract-based access control through off-chain signature and on-chain evaluation," *IEEE Trans. Circ. Syst. II: Express Briefs*, vol. 69, no. 4, pp. 2221–2225, 2021. doi: 10.1109/TCSII.2021.3125500.

[22] P. Zhu, C. Miao, Z. Wang, and X. Li, "Informational cascade, regulatory focus and purchase intention in online flash shopping," *Electron. Commer. Res. Appl.*, vol. 62, pp. 101343, 2023. doi: 10.1016/j.elerap.2023.101343.

[23] P. Zhu, Z. Liu, X. Li, X. Jiang, and M. X. Zhu, "The influences of livestreaming on online purchase intention: Examining platform characteristics and consumer psychology," *Ind. Manage. Data Syst.*, vol. 123, no. 3, pp. 862–885, 2023. doi: 10.1108/IMDS-07-2022-0430.

[24] H. Kaur, M. A. Alam, R. Jameel, A. K. Mourya, and V. Chang, "A proposed solution and future direction for blockchain-based heterogeneous medicare data in cloud environment," *J. Med. Syst.*, vol. 42, pp. 1–11, 2018. doi: 10.1007/s10916-018-1007-5.

[25] C. Zhang, C. Hu, T. Wu, L. Zhu, and X. Liu, "Achieving efficient and privacy-preserving neural network training and prediction in cloud environments," *IEEE Trans. Dependable Secur. Comput.*, vol. 20, no. 5, pp. 4245–4257, 2023. doi: 10.1109/TDSC.2022.3208706.

[26] C. Zhang, X. Luo, J. Liang, X. Liu, L. Zhu, and S. Guo, "POTA: Privacy-preserving online multi-task assignment with path planning," *IEEE Trans. Mob. Comput.*, pp. 1–13, 2023. doi: 10.1109/TMC.2023.3315324.

[27] C. Zhang, M. Zhao, J. Liang, Q. Fan, L. Zhu, and S. Guo, "NANO: Cryptographic enforcement of readability and editability governance in blockchain database," *IEEE Trans. Dependable Secur. Comput.*, pp. 1–14, 2023. doi: 10.1109/TDSC.2023.3330171.

[28] J. Wang, J. Yu, Y. Song, X. He, and Y. Song, "An efficient energy-aware and service quality improvement strategy applied in cloud computing," *Clust. Comput.*, vol. 26, no. 6, pp. 4031–4049, 2023. doi: 10.1007/s10586-022-03795-w.

[29] J. Wang, J. Yu, R. Zhai, X. He, and Y. Song, "GMPR: A two-phase heuristic algorithm for virtual machine placement in large-scale cloud data centers," *IEEE Syst. J.*, vol. 17, no. 1, pp. 1419–1430, Mar. 2023. doi: 10.1109/JSYST.2022.3187971.

[30] J. Wang, H. Gu, J. Yu, Y. Song, X. He, and Y. Song, "Research on virtual machine consolidation strategy based on combined prediction and energy-aware in cloud computing platform," *J. Cloud Comput.*, vol. 11, no. 1, pp. 1–18, 2022. doi: 10.1186/s13677-022-00309-2.
[31] G. Wang, S. Zhang, T. Yu, and Y. Ning, "A systematic overview of blockchain research," *J. Syst. Sci. Inf.*, vol. 9, no. 3, pp. 205–238, 2021. doi: 10.21078/JSSI-2021-205-34.
[32] F. Casino, T. K. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications: Current status, classification and open issues," *Telematics and Informatics*, vol. 36, pp. 5581, 2019. doi: 10.1016/j.tele.2018.11.006.
[33] E. Androulaki *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. Thirteen. EuroSys Conf.*, 2018, pp. 1–15.
[34] H. Z. Li, C. X. Li, H. X. Li, X. Q. Bai, and X. Shi, "An overview on practice of FISCO BCOS technology and application," *Inf. Commun. Technol. Policy*, vol. 46, no. 1, pp. 52–60, 2020. http://ictp.caict.ac.cn/EN/Y2020/V46/I1/52
[35] M. Alharby, A. Aldweesh, and A. v. Moorsel, "Blockchain-based smart contracts: A systematic mapping study of academic research," in *2018 Int. Conf. Cloud Comput., Big Data and Blockchain (ICCBB)*, Fuzhou, China, 2018, pp. 1–6.
[36] S. Ghemawat, H. Gobioff, and S. T. Leung, "The google file system," in *Proc. 19th ACM Symp. Oper. Syst. Princ.*, Bolton Landing, NY, 2003, pp. 20–43.
[37] Q. Xu, R. V. Arumugam, K. L. Yang, and S. Mahadevan, "DROP: Facilitating distributed metadata management in EB-scale storage systems," in *2013 IEEE 29th Symp. Mass Storage Syst. Technol. (MSST)*, Long Beach, CA, USA, 2013, pp. 1–10.
[38] J. Zhou, Y. Chen, M. Zheng, and W. Wang, "Data distribution for heterogeneous storage systems," *IEEE Trans. Comput.*, vol. 72, no. 6, pp. 1747–1762, 2023. doi: 10.1109/TC.2022.3223302.
[39] A. Mushtaq, "Fault tolerance in distributed systems," Ph.D. dissertation, Univ. of California, Berkeley, May 2022.
[40] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proc. 7th Symp. Oper. Syst. Des. Implement.*, Seattle, WA, USA, 2006, pp. 307–320.
[41] D. Borthakur, "HDFS architecture guide," *Hadoop Apache Project*, vol. 53, no. 1–13, pp. 2, 2008.
[42] P. Braam, "The Lustre storage architecture," arXiv preprint arXiv:1903.01955, 2019.
[43] J. Benet, "IPFS-Content addressed, versioned, P2P file system," arXiv preprint arXiv:1407.3561, 2014.
[44] F. W. Chang *et al.*, "Bigtable: A distributed storage system for structured data," in *Operating Systems Design and Implementation*, New York: Association for Computing Machinery, 2008. doi: 10.1145/1365815.1365816.
[45] Z. Huang, H. Jiang, K. Zhou, Y. Zhao, and C. Wang, "Lowest density MDS array codes of distance 3," *IEEE Commun. Lett.*, vol. 19, no. 10, pp. 1670–1673, Oct. 2015. doi: 10.1109/LCOMM.2015.2464379.
[46] Z. Huang, H. Jiang, K. Zhou, C. Wang, and Y. Zhao, "XI-code: A family of practical lowest density MDS array codes of distance 4," *IEEE Trans. Commun.*, vol. 64, no. 7, pp. 2707–2718, Jul. 2016. doi: 10.1109/TCOMM.2016.2568205.
[47] Z. Huang, H. Jiang, and N. Xiao, "Efficient lowest density MDS array codes of column distance 4," in *2017 IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 834–838.
[48] Z. Huang, H. Jiang, H. Che, N. Xiao, and N. Li, "Efficient MDS array codes for correcting multiple column erasures," in *2019 IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 1602–1606.
[49] Z. Huang, H. Jiang, Z. Shen, H. Che, N. Xiao, and N. Li, "Optimal encoding and decoding algorithms for the RAID-6 liberation codes," in *2020 IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, New Orleans, LA, USA, 2020, pp. 708–717.
[50] Z. Huang, H. Jiang, and K. Zhou, "An improved decoding algorithm for generalized RDP codes," *IEEE Commun. Lett.*, vol. 20, no. 4, pp. 632–635, Apr. 2016. doi: 10.1109/LCOMM.2016.2522414.