Article

CMSS: A High-Performance Blockchain Storage System with Horizontal Scaling Support

Wenjin Yang [1,2], Meng Ao [2], Mingzhi Gao [1], Chunhai Li [3,*] and Yongqing Chen [2,*]

[1] School of Cyberspace Science & Technology, Beijing Institute of Technology, Beijing 100081, China; wenjinyang@bit.edu.cn (W.Y.); gaomingzhi@bit.edu.cn (M.G.)
[2] Tencent Inc., Shenzhen 518055, China; mengao@tencent.com
[3] Guangxi Engineering Research Center of Industrial Internet Security and Blockchain, Guilin University of Electronic Technology, Guilin 541004, China
* Correspondence: chunhaili@guet.edu.cn (C.L.); hokchen@tencent.com (Y.C.)

Abstract: As a decentralized system, blockchain has been widely used in numerous areas and has become a hot topic in both industry and academia. The increasing demand for blockchain causes heavy storage consumption which seriously affects the performance of blockchain, especially in the context of massive volumes of data. To solve these problems, many related systems like sharding and sidechain have been proposed to improve the efficiency and scalability of blockchain. However, in practical deployment scenarios, these systems still have problems, such as low read–write performance, and reorganization and synchronization of ledger data after storage expansion, which cause the storage system's expansion to become difficult and time-consuming in large-scale blockchain systems. Facing these problems, in this paper we propose **C**hain**M**aker **S**torage **S**ystem (CMSS). CMSS is a blockchain storage system with high read-and-write performance and horizontal scaling support. It has been used as the storage system of the most popular permissioned blockchain ChainMaker. There are three contributions of our proposed CMSS: (i) a new block storage workflow to achieve high read–write performance; (ii) the **M**eta **F**ile **S**ystem (MFS) to support the horizontal expansion of blockchain storage; and (iii) hot–cold separation to reduce the resource usage and economic costs. To evaluate the performance of CMSS, we compare CMSS with Hyperledger Fabric (HLF), the most popular permissioned blockchain platform. We select five well-known cloud service providers to calculate the storage cost in a real production environment. The results show that CMSS has better performance in read and write than HLF and advantages in storage capacity and price.

Keywords: blockchain storage; system scalability; permissioned blockchain

check for updates

Citation: Yang, W.; Ao, M.; Gao, M.; Li, C.; Chen, Y. CMSS: A High-Performance Blockchain Storage System with Horizontal Scaling Support. *Electronics* **2024**, *13*, 1854. https://doi.org/10.3390/electronics13101854

Academic Editor: Fabio Grandi

Received: 7 April 2024
Revised: 5 May 2024
Accepted: 6 May 2024
Published: 10 May 2024

1. Introduction

The blockchain system was first proposed in 2008 [1] as a solution to prevent tampering and double record issues in the peer-to-peer version of the electronic cash system [1]. With the support of cryptographic algorithms [2] and the increasing emphasis on data privacy and data rights confirmation [3], blockchain, one of the most popular decentralized systems, has been used as the underlying architecture in numerous applications like non-fungible tokens (NFTs) [4] and cryptocurrency [5]. Blockchain technology presents a trilemma of decentralization, security, and scalability, wherein the pursuit of security and decentralization invariably compromises scalability. To ensure the safety of users' property, both Bitcoin [1] and Ethereum [6] prioritize decentralization and security at the expense of scalability. Scalability has become the biggest bottleneck in the implementation of blockchain applications. For example, the soaring number of accounts and transactions made the Ethereum data more than 16.5 TB as of 18 February 2024 [7] and the storage size of Bitcoin [1] is 549.46 GB on 15 February 2024 [8]. To maintain the consistent order of transactions, ensure the integrity of the execution process, and support the traceability of data sources, current blockchain technologies require that every node in the network should possess a comprehensive replication of the complete transaction history and ledger state, which seriously influences the performance of blockchain in latency and throughput. Facing this problem, many scholars have proposed many solutions to minimize the storage pressure of blockchain and improve the performance and storage scalability of blockchain. Sharding, proposed by [9–11], can divide the entire blockchain network into multiple shards. The nodes in each shard are responsible for processing the transactions of the shard and storing the status of the shard. Sharding technology can achieve the high-performance goals of the blockchain without sacrificing the degree of centralization and has gradually become one of the mainstream blockchain expansion technologies. However, the existing sharding technology still leads to new problems, like cross-shard transaction processing and attacks by slowly adaptive Byzantine adversaries [9]. Sidechain, proposed as another solution [12,13], is an independent blockchain that can run in parallel with the main chain and supports the cross-chain transmission of assets. Sidechain can reduce the burden of the mainchain and improve scalability. The above solutions still face the problem that the file systems of blockchain cannot expand horizontally. When the disks of file system are full, the chain stops working and cannot store new blocks. When meeting massive data, the blockchain will become cumbersome, non-scalable, and slow to operate. That means the blockchain needs to reorganize and synchronize ledger data after the storage expansion.

After analyzing the problems and solutions in the blockchain storage system, we summarize the following research questions worth attention and research. These research questions are also the starting point of our work.

RQ1 **How to achieve efficient blockchain operations in scenarios with massive ledger data, such as reading and writing block operations?** When the amount of ledger data is large, the operation of reading and writing blocks becomes time-consuming, which causes delays and affects the throughput of the entire blockchain system, which becomes one of the bottlenecks of the system.

RQ2 **How to improve the performance when adding more disks without reorganizing the ledger data?** The current technology writes the block data on the chain into a file system. The upper limit of block storage in the blockchain is limited by the size of the file system. When the file system (disk) is full, the chain stops working and cannot store new blocks. If we add more disks to this system, the blockchain reorganizes the ledger data and re-download the ledger data from the network. This is an extremely time-consuming process.

RQ3 **As the scale of blockchain ledger data continues to expand, how to control the storage cost of the entire blockchain?** With the development of blockchain systems, the amount of ledger data will become very large. Each node in the blockchain system needs to save all ledger data, which will bring a huge storage burden and expensive storage costs to the entire system.

To solve these research questions, we propose CMSS, **C**hain**M**aker **S**torage **S**ystem, in this work. CMSS is a blockchain storage system with high read-and-write performance and horizontal scaling support. CMSS has a new workflow and architecture. It has been used as the storage system of the most popular permissioned blockchain ChainMaker [14]. The new workflow of CMSS can achieve high performance in reading and writing data while ensuring the consistency of the ledger. The architecture of CMSS supports rapid horizontal scalability based on the **m**eta **f**ile **s**ystem (MFS) we proposed. To minimize the consumption of storage and reduce the economic costs, our proposed CMSS provides hot–cold separation supports. In the experiment, we compare CMSS with the storage system of state-of-the-art permissioned blockchains such as Hyperledger Fabric (HLF). The results of the experiments show that our storage system has significant advantages in latency and throughput. Especially in scenarios of massive data, our storage system has advantages over HLF in terms of efficiency and cost-effectiveness.

The major contributions are summarized below.

1. **New Workflow with High Read–Write Performance**. We propose a new block storage workflow. Following this process, CMSS can improve the reading and writing efficiency of the blockchain system while ensuring the consistency of the ledger.
2. **MetaFile System with Horizontal Scalability**. We creatively propose a file architecture for the blockchain storage system that supports rapid horizontal scalability. When meeting scenarios with massive data, this architecture can expand ledger capacity by only adding more disks without reorganizing the data.
3. **Hot–Cold Separation with Lower Storage Costs**. To save on the storage hardware costs, we have designed a block storage management solution based on hot and cold separation. Experiments prove that our blockchain storage system has lower costs than other permissioned blockchains.

The remainder of this paper is structured as follows. Section 2 reviews the related work in this area. Section 3 provides the necessary background and problem settings. Section 4 introduces the architecture of CMSS and the new workflow. Section 5 explains the features of CMSS and its underlying logic. Section 6 contains the implementation and a series of experiments. Section 7 concludes this paper.

2. Related Work

In this section, we introduce the current technology used in the storage system of blockchain to minimize storage pressure and improve performance and storage scalability.

Sharding. The limited scalability of blockchain technology has hindered its development and application. Recently, sharding has been employed in blockchain systems to partition the network into multiple committees that can process transactions concurrently. Sharding divides the entire blockchain network into different shards, and the nodes in each shard are responsible for processing the transactions of the shard and storing the status of the shard. Initially, shared blockchain systems only supported network and transaction sharding based on the UTXO model, such as Elastico [15] and Monoxide [16]. For instance, in Elastico, each block needs to be broadcast to all nodes for storage. However, more recent systems have started to incorporate state sharding in blockchain, such as Chainspace [17] and AHL [10], which generalize the sharding technique to smart contract applications beyond cryptocurrency. AHL [10] proposes a distributed transaction protocol for smart contracts based on two-phase commit (2PC) and two-phase locking (2PL) protocols to address cross-committee transactions that manipulate states in multiple committees. Hellings et al. [18] proposed ByShard, a unifying framework for the study of shared resilient systems, to provide a solid foundation for the development of ACID-compliant general-purpose and flexible shared resilient data management systems. GRIDB, proposed by Hong et al. [19], is a scalable blockchain database with a novel off-chain cross-shard mechanism for efficiency cross-shard database services. GRIDB achieves a scalable throughput for SQL linearly increasing with the shard number compared with the non-sharding systems.

Sidechain. Sidechain was first introduced by Back et al. [20]. A notable implementation of sidechains is Drivechains [21], which employs simple payment verification (SPV) proofs [22] to facilitate transfers from the mainchain to sidechains. However, the proof size of SPV proofs is linear in the chain length, which can be large and unwieldy. To address this issue, PoW sidechains [12] were proposed, achieving logarithmic proof size. While this construction is suitable for PoW blockchains with a fixed difficulty, it does not apply to blockchains with variable difficulty. Subsequently, the FlyClient protocol [23] was introduced for verifying blockchains with variable difficulty. Nevertheless, both the PoW sidechain construction and FlyClient are designed specifically for the PoW setting. PoS sidechains, a sidechain construction that is suitable for proof-of-stake (PoS) sidechain systems, was proposed by Gaži et al. [12]. In 2023, Yin et al. [24] proposed a generic sidechain construction named Ge-Co to realize secure asset transfers between blockchains. Ge-Co supports different consensus algorithms, such as proof of stake (PoS) and proof of work (PoW). Deng et al. [25] proposed practical and secure sidechain construction (PSSC) in the form of parent–child chains, a practical and secure sidechain construction for heterogeneous blockchains orienting IoT scenarios.

BFT-Store. To mitigate the negative impacts of the full-replication strategy while ensuring the availability of all blocks, a reliable storage scheme called BFT-Store has been recently proposed for permissioned blockchain systems [26]. This innovative approach aims to address the scalability challenges faced by blockchain technology, enabling efficient and secure data storage in a decentralized manner. However, BFT-Store still has some problems [27], like complex re-initialization, high computational complexity, and massive communication on the network.

Other techniques. Bagozi et al. [28] presented an approach that encompasses criteria for identifying trust-demanding objects and activities within collaborative processes. Additionally, they introduced a model to describe smart contracts in a platform-independent manner, along with guidelines for their deployment on a blockchain. After prior works, they [29] introduced a methodology and a tool that utilizes step-by-step procedures to facilitate trust management in web-based collaborative business processes. These processes were initially designed based on a centralized BPM strategy but are now adapted for blockchain integration.

All of these works enhance blockchain scalability while ensuring transaction atomicity and consistency. However, they still face challenges such as heavy storage consumption and economic costs. In practical deployment scenarios, when storage becomes depleted, these proposed solutions require reorganization and synchronization of ledger data after storage expansion. In response to these challenges, we introduce CMSS in this work to achieve high efficiency, minimize storage consumption, and reduce economic costs.

3. Preliminaries

In this section, we provide the necessary background, such as the regular blockchain workflow, the architecture of the normal blockchain storage system, and transaction deduplication in permissioned blockchain.

3.1. Blockchain Workflow

Based on its access control mechanism and permission management methods, blockchain can be divided into two categories: permissionless blockchain such as Bitcoin [1] and Ethereum [6], and permissioned blockchain such as Hyperledger Fabric [30]. Similar to permissionless blockchains, permissioned blockchains also receive transactions from clients, go through a workflow to update the ledger, and ensure consistency among replicas in a trustless environment. Based on their workflows, permissioned blockchains can be broadly classified into two distinct categories.

Execute $\rightarrow$ Order $\rightarrow$ Validate Workflow. Hyperledger Fabric is the most traditional example following this workflow. In this workflow, transactions are executed concurrently first, and then, the order of transaction submission is determined. Transactions can be executed concurrently across nodes during the execution phase. After using the sorting service to determine the global order and package it into blocks, the blocks are sent to all nodes for serial read-and-write conflict verification.

Order $\rightarrow$ Execute Workflow. Quorum [31] leverages this workflow to process transactions. In this workflow, nodes initially reach a consensus on the order of transactions. Subsequently, each node executes transactions according to the determined sequence. Consequently, transactions are less likely to be aborted due to contentions.

3.2. Storage System

The data storage system of the Hyperledger Fabric blockchain system mainly consists of one file storage (block data) and three databases. The structure is shown in Figure 1.

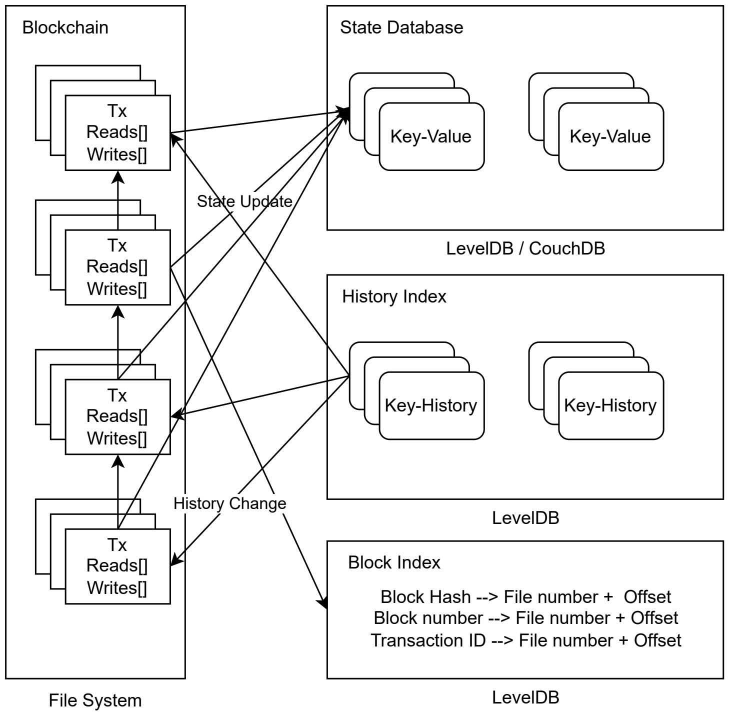


Figure 1. The storage system of Hyperledger Fabric.

The structure of the Hyperledger Fabric storage system consists of four parts:

- **File System/Ledger**. The file system (also known as the ledger) is used to store all the transaction data in a sequential and immutable manner. It contains a series of blocks linked by cryptographic hashes [32] and each block contains a set of transactions. Each peer in the network maintains a copy of the ledger to ensure decentralization and resilience. The ledger is stored as a set of files on disk, with each block being stored in a separate file.
- **State Database**. The state database maintains the current state of the ledger, which represents the most recent value of each key–value pair in the system. It is essentially a key–value store that allows for fast retrieval of the latest state information without having to traverse the entire blockchain. The state database is critical for performance, as it enables applications to query the current state of the ledger without needing to process all previous transactions. Hyperledger Fabric supports multiple state database implementations, including LevelDB [33] and CouchDB [34].
- **History Index**. The history index keeps track of the historical values of keys in the state database. It allows users to query the historical state of a key at any point in time. This feature is particularly useful for auditability. The history index is implemented as a separate key–value store, with keys being the historical versions of the state database keys and values being the corresponding historical values.
- **Block Index**. Hyperledger Fabric provides a variety of indexing methods to quickly find the required block data. The index database is updated every time a block is submitted. It can efficiently locate blocks in the ledger based on certain criteria, such as block number or transaction ID. It is essentially a mapping of block metadata to block file locations on disk.

3.3. Sharding

Sharding [35], derived from database partitioning, involves breaking down a large database or blockchain into smaller, manageable parts called shards. This enables parallel processing of transactions, thereby enhancing scalability. In blockchain systems, sharding divides nodes and the ledger into shards, allowing for distributed processing of transactions across shards. Here are the key components of the sharding protocol:

- **Shard Formation**: Nodes establish identities using Sybil-attack-resistant [36] methods like proof of work (PoW). Each node is then randomly assigned to a shard to ensure honesty with high probability. Additionally, shards need periodic reconfiguration to prevent attacks.
- **Cross-Shard Mechanism**: Since the ledger is divided into shards, cross-shard transactions occur frequently. A cross-shard mechanism is necessary to ensure the atomicity (transactions are committed or aborted as a whole) and consistency (each transaction commit leads to a valid state) of these transactions across shards.
- **Intra-Shard Consensus**: Within each shard, nodes must reach consensus on a block containing proposed transactions. A Byzantine consensus protocol is typically employed to ensure safety (honest nodes agree on the same value) and liveness (valid transactions are eventually included in the ledger).

3.4. Sidechains

Sidechain, also called pegged sidechains, is a cross-chain technique that enables blockchain interoperability through sidechains construction [20]. Sidechain supports the transfer of digital assets or data between different blockchains. Users send assets to an external address linked to a federation to move assets from the mainchain to the sidechain. This federation acts as a mediator, facilitating the locking and unlocking of assets or data between the mainchain and the sidechain. After a specified transaction period, the federation releases equivalent assets on the sidechain. Sidechains typically exist in two forms: parallel chains and parent–child chains. In the parent–child chain design, the mainchain serves as the parent chain, while additional chains function as sidechains.

4. Design of CMSS

In this section, we propose the design of our **C**hain**M**aker **S**torage **S**ystem (CMSS). First, we introduce the system model, and then, we define the new workflow of our system.

4.1. System Model

The storage system is responsible for persisting ledger data such as blocks, transactions, statuses, and historical read-and-write sets on the storage chain, and providing external query functions for the above data. In this study, we focus on the file system level to build a blockchain storage system that supports rapid horizontal scalability. The architecture is shown in Figure 2.

Ledger with multiple file systems. To solve the problem that the traditional ledger is difficult to expand, we establish a new structure at the file system level. The new ledger can be composed of multiple file systems, each file system contains multiple block files, and each block file stores multiple transactions. In our scheme, blockchain supports setting multiple file systems. While the system is running, the blockchain also supports the continuous addition of file systems (disks) to achieve horizontal expansion and expand the upper limit of ledger-stored data.

Cache that supports high-frequency contracts. For smart contracts with high-frequency reading and writing characteristics, we design a cache component to optimize their reading and writing operations. The database itself has an internal cache that can enhance the reading and writing efficiency of the database. However, as the volume of data in the database grows, the internal cache's ability to improve database read-and-write efficiency becomes limited. To address this issue, our architecture incorporates an additional layer of cache that is separate from the database. This isolated cache can maximize read-and-write efficiency, particularly in situations involving large amounts of data. By adding this separate cache layer, we can overcome the limitations of the internal cache and ensure optimal performance even as the volume of data continues to increase.

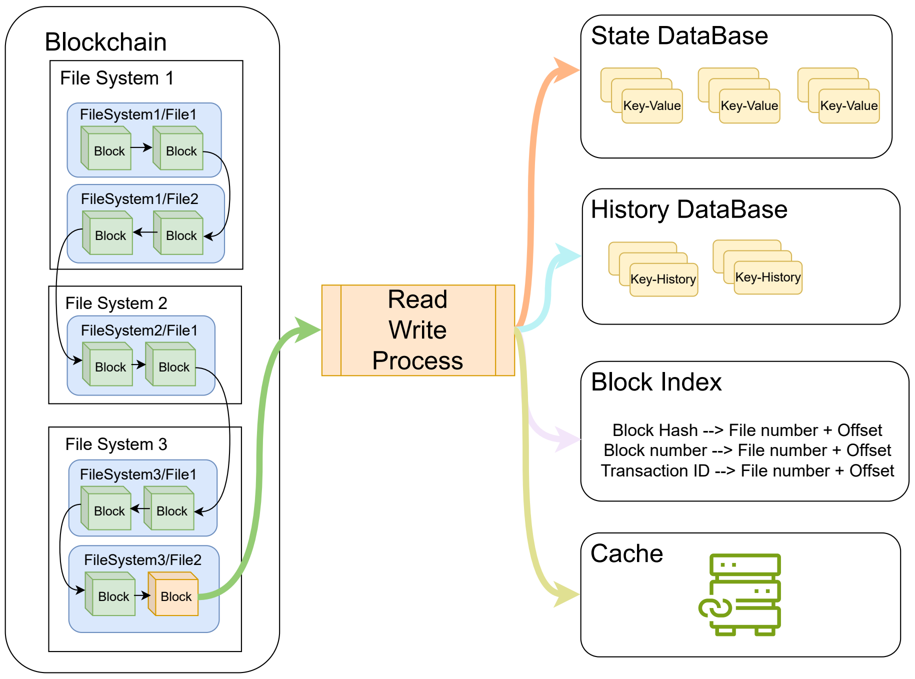


Figure 2. The system model of CMSS.

4.2. CMSS's Workflow

Now, we describe the workflow of our proposed architecture, which aims to optimize the performance of the blockchain storage system. The workflow can be divided into six main steps and is shown in Figure 3.

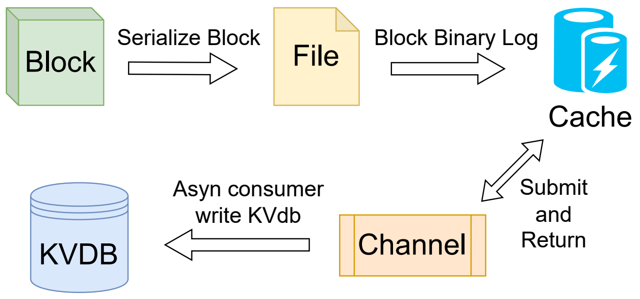


Figure 3. The workflow of CMSS.

1. **Serialization**: Parallel serialization of new blocks.
2. **Block Binary Log Writing**: Write the read–write sets, the latest block height to the block binary log. This step is essential for data recovery in case of abnormal interruptions.
3. **Caching**: In order to improve performance, a layer of cache is added. After the new block submission request updates the block binary log, the block data are written into the cache.
4. **Submit and Return**: Upon completion of both log and cache updates, the commit request can return, and the background thread asynchronously updates the relevant databases.
5. **Asynchronous Database Update**: The background thread updates the Block DB, State DB, ContractEvent DB, History DB, and Result DB asynchronously, ensuring optimal performance and resilience against failures.

By following this workflow, our proposed architecture ensures efficient data processing, resilience against failures, and optimal performance, making it suitable for various blockchain-based applications in enterprise environments.

5. System Features

In this section, we introduce the system features of our CMSS (**C**hain**M**aker **S**torage **S**ystem). As a critical component of the blockchain ecosystem, the storage system is responsible for securely and efficiently storing and managing the vast amount of data generated by blockchain transactions. Our proposed CMSS incorporates several key features that enhance its performance, scalability, and security, ensuring that it meets the unique requirements of blockchain-based applications. These features include high-performance read–write ability, horizontal scaling, transactions deduplication, and hot–cold separation and archiving.

5.1. High-Performance Read–Write Ability

Most of the existing blockchain storage systems, such as LevelDB and CouchDB, use a key–value database as their underlying architecture. Although there is a cache inside the key–value database to improve reading and writing efficiency, in a massive-data scenario, the amount of data stored in the database is large, and the internal cache has little improvement in reading and writing efficiency, and there is even a possibility of cache failure.

In our proposed architecture, we add a cache component that is isolated from the database. This cache only stores the smart contracts that are often accessed and the stored smart contracts can be changed by LRU or LFU algorithm. Using this cache component, the CMSS can achieve high performance in reading and writing operations. When writing blocks, our CMSS writes blocks to a key–value database and cache, which enhances reading performance. During the reading process, CMSS efficiently retrieves data directly from the cache.

5.2. Horizontal Scaling

To achieve horizontal scaling, we need to analyze the relationship between blocks, files, file systems, and the ledger. In blockchain, a block is written into a file after serialization. The file belongs to a file system. The relationship between them can be shown as follows:

- one ledger has many file systems;
- one file system has many files;
- one file stores many blocks.

5.2.1. Meta File System

To manage the relationships above, we propose the **m**eta **f**ile **s**ystem (MFS) that supports our CMSS, giving it the ability of horizontal scaling. The key features of our proposed system include the ability to retrieve the file system and filename associated with a specific block height, as well as to provide the last used file system and filename. Furthermore, our system supports at least two file system switching strategies and allows for an arbitrary number of file systems to be added.

5.2.2. Architecture of MFS

The architecture of our proposed MFS is shown in Figure 4. The MFS consists of three parts: MetaIndex, setting, and database.

MetaIndex. The metadata system needs to find out the file name and file system information corresponding to this block based on the block height. With a huge number of blocks, high search performance is required. If you use traversal search, the time complexity is O(n). Through a B-Tree, with the block height as the key and the leaf node as file metadata, the file metadata information corresponding to the corresponding block height can be found efficiently. It can support 1 PB of data, the time complexity is O(m), and m is generally less than 10.

Setting. To support horizontal scaling, MFS needs to keep the online file system in the setting. MFS provides multi-version control settings and can persist historical settings.

Every time the user modifies the configuration, the new configuration needs to be saved in the metadata system and persisted.

Database. It is mainly a key–value structure data, with the file name as the key and value as the serialized data of file metadata. It is used to store metadata information persistently in the DB. When the process is restarted, metadata information is read from the database and index data are constructed. Metadata information needs to be persisted. After the server or process is restarted, the metadata information is read from the persistent data and the index information is regenerated in memory. Persistence uses LevelDB to save data, the key is the file name, and the value is the serialized data of fileMeta.

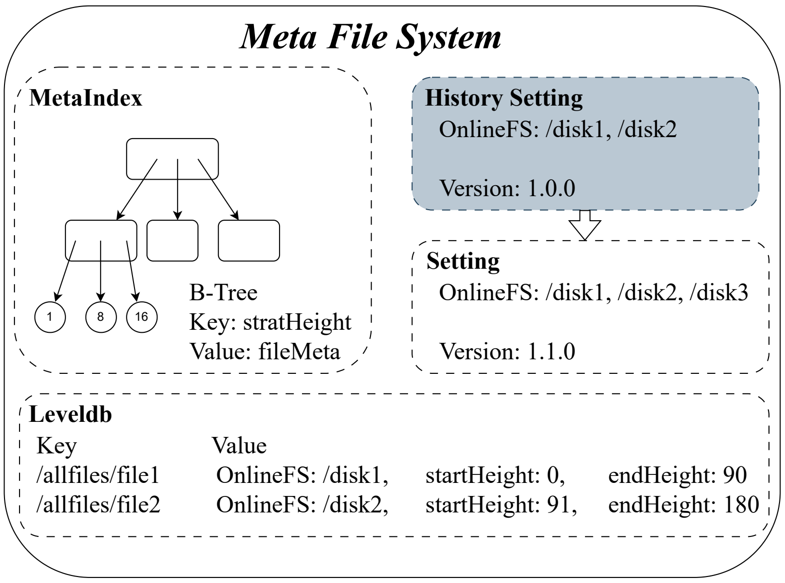

Meta File System
MetaIndex
1
8
16
B-Tree
Key: stratHeight
Value: fileMeta
History Setting
OnlineFS: /disk1, /disk2
Version: 1.0.0
Setting
OnlineFS: /disk1, /disk2, /disk3
Version: 1.1.0
Leveldb
Key
Value
/allfiles/file1 OnlineFS: /disk1, startHeight: 0, endHeight: 90
/allfiles/file2 OnlineFS: /disk2, startHeight: 91, endHeight: 180

Figure 4. The architecture of MFS.

With the support of MFS, CMSS can achieve horizontal expansion on the ledger data storage without the need to reorganize the data after the ledger storage expansion is completed.

5.3. Hot–Cold Separation

In blockchain technology, it has been observed that transactions predominantly accessed tend to be those that have been recently inscribed, while older transactions, such as those dating back several months or even years, experience a diminished frequency of access. As the accumulation of transactions persists and the generation of new blocks continues, there is a consequential increase in the number of files and the storage space they occupy. This phenomenon leads to the emergence of several notable trends: firstly, the total volume of transactions experiences a gradual growth over time; secondly, storage costs witness a concomitant escalation; and thirdly, the quantity of highly accessed transactions does not exhibit a proportional increase to the overall transaction volume.

To solve these problems, we propose the hot–cold separation. Our technology delivers solutions that balance cost, performance, and functionality while ensuring atomicity, consistency, tamper resistance, and recoverability. The process of the hot–cold separation is shown in Figure 5.

In Figure 5, the user initiates separation and reviews separation results asynchronously in step 1. Then, the separation task is produced and sent into the separation queue in step 2. Asynchronously, the separation task is written into the database. Step 3 consumes a task from the separation queue. The separation automaton is called and the separation executed in step 4. Separation automation is introduced in the next paragraph. After separation, the result updates the separation state in the database. When a failure occurs during the separation process, the separation operation resumes at the breakpoint. The unfinished separation tasks are scanned out from the database and re-sent to the separation queue to continue to complete the separation.

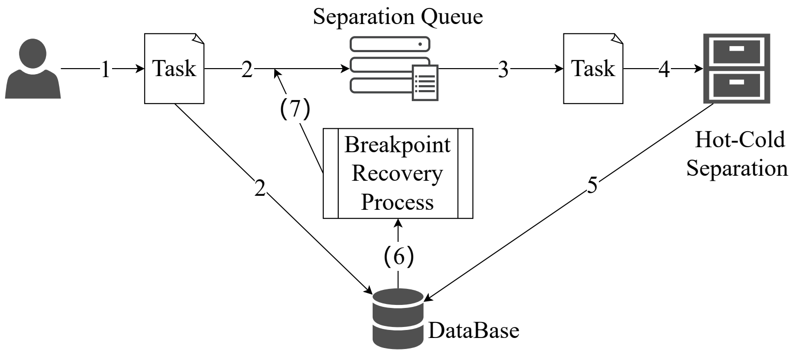


Figure 5. The process of hot–cold separation.

Separation Automation. Separation automation includes a separation task status machine and a file status machine. In the process of separation, the separation task status machine is called once and the file status machine is called N times (N is the number of files). The design of separation automation is shown in Figure 6.

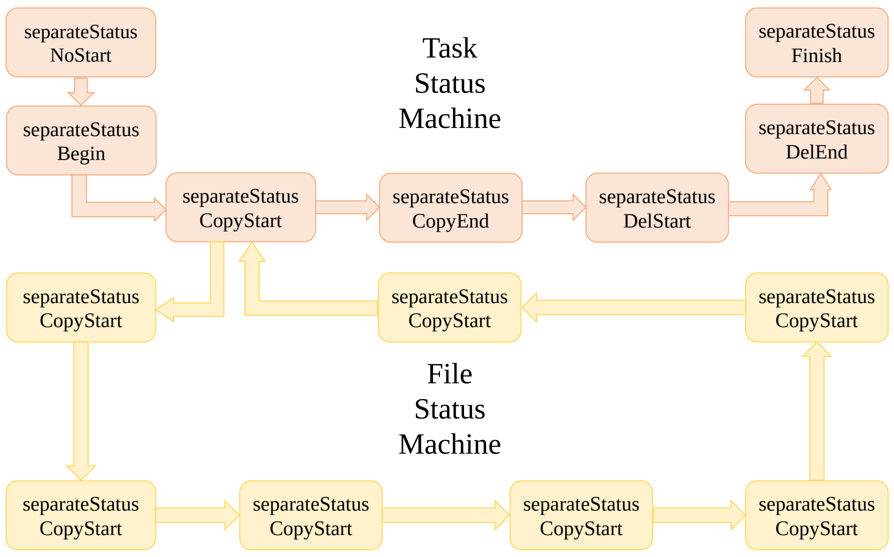


Figure 6. The separation automation.

Breakpoint Recovery. During the separation process, both the separation task status machine and the file status machine are employed. Each time an operation is executed, it is initially performed in the current status. Upon completion of the execution, the task status is updated in the database to ensure the persistence of the status. Consequently, in the event of a failure, it is possible to resume the process by relying on the pre-existing state before the failure and continue accordingly.

6. Experiment

To evaluate the performance of our proposed ChainMaker Storage System, we set a series of experiments that follow the research questions proposed in Section 1. Here, we introduce the experiment setup and other components that are necessary for evaluation and comparison. By analyzing the experiment data and logic of the experiments, we demonstrate that our protocol meets expectations.

6.1. Experiment Setup

We first introduce the basic information about our experiment setup.

Baseline. To better prove the effectiveness of our CMSS, we chose Hyperledger Fabric [30], the most popular enterprise-grade permissioned blockchain platform that is used worldwide in numerous scenarios.

Testbed. We deploy Hyperledger Fabric and our CMSS in Tencent Cloud environments for our experiments. The compute service instance from Tencent Cloud is equipped with an AMD EPYC 7K62 48-Core Processor, 32 GB of RAM, 150 GB Disk, and TencentOS (Linux based). Especially, to meet the storage needs of massive data, we apply a 1000 GB cloud block storage (CBS) for our instance.

Implements. To better compare the performance of storage systems and discard the impact of other components (such as consensus [37,38], transaction scheduling [39], etc.) on performance, we isolated the code of Hyperledger Fabric and CMSS and only selected the storage system of these two permissioned blockchains for comparison.

Evaluation. We use QPS and average time in our experiment to evaluate the performance of our CMSS. QPS stands for queries per second and is a metric used to measure the number of queries or requests processed by a system within one second. QPS is commonly used in the context of databases, web servers, and other systems that handle a large number of requests. A higher QPS indicates better performance and scalability of the system, as it shows the system's ability to efficiently handle a large volume of requests within a given time frame.

$$QPS = \frac{Total\ number\ of\ queries}{Time\ period(in\ seconds)} \tag{1}$$

6.2. Read–Write Performance Comparison between CMSS and HLF

Based on the current user usage scenarios of ChainMaker, we classify user contracts into three categories according to the types of storage read-and-write operations: certificate type [40], NFT type [4], and batch creation type (TDID [41]). These three types of contract have their own characteristics from a storage perspective:

Certificate Type: These contracts are one-time deployed onto the blockchain and almost no read operations are performed. The read-to-write ratio is typically 0:1.

NFT Type: These contracts facilitate various operations such as the issuance, sale, and transfer of NFTs. They involve a significant volume of both read and write operations.

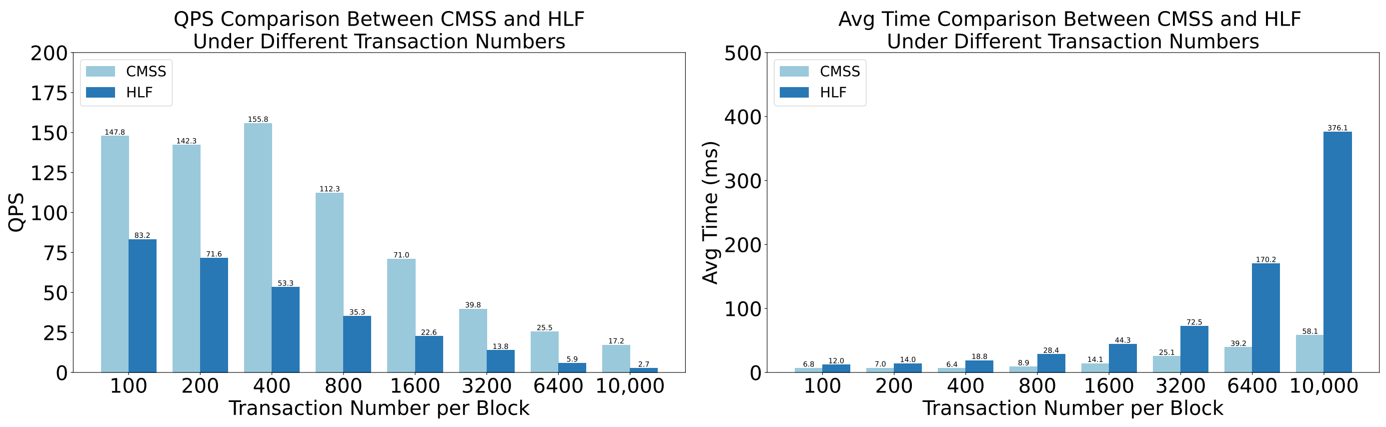
Batch Creation Type: These contracts are responsible for generating multiple key–value data pairs, often involving read operations on the keys. Each transaction typically contains several key–value pairs.

Based on this classification, we build three types of experiments: write, key-exist read last written, key-exist random read, and read key-not-exist.

6.2.1. Write

We conducted experiments using a producer–consumer model to assess our storage system. Specifically, the producer is responsible for block creation, initializing the number of transactions in each block to 100, 200, 400, 800, 1600, 3200, 6400, and 10,000. Using random functions, the producer populates each transaction with corresponding read–write sets and state data. Subsequently, the producer writes the generated blocks into a channel. The consumer, on the other hand, cyclically retrieves blocks and read–write sets from the channel and subsequently writes them into the storage system.

Our proposed CMSS achieves higher throughput and lower latency than HLF, as shown in Figure 7. Figure 7a illustrates the QPS of the system and Figure 7b displays the average time in milliseconds. Figure 7a shows that our CMSS has a better QPS performance. In the situation of 1000+ transactions in each block, our CMSS still improves the QPS by 3×. In particular, when the transaction number of each block is 10 thousand, our CMSS improves the QPS by 5.3×. While having higher throughput, our CMSS also has lower time consumption compared to HLF. Our CMSS has smaller times than HLF both when the number of transactions in the block is low (100 transactions each block; improvement of 1×) and when the block load is large (10,000 transactions each block; improvement of 5.5×).

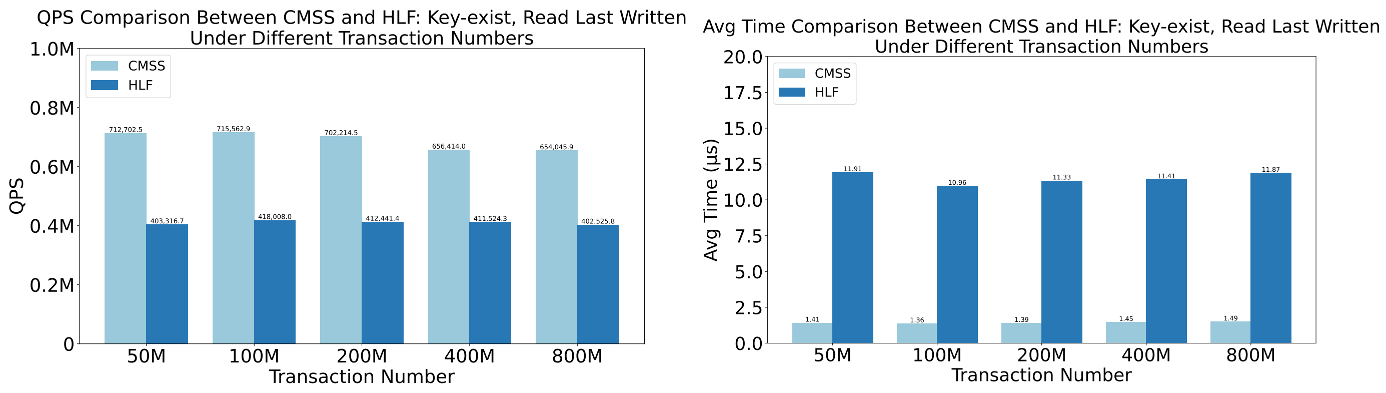


(**a**) QPS comparison in write experiment (**b**) Average time comparison in write experiment

Figure 7. The experimental data from write comparison.

6.2.2. Key-Exist Read Last Written

In this experiment, we require that all read operations must satisfy that the content being queried has been written recently (for example, we only query the content in the last 100 blocks). In this case, the queried content is hit in the cache, and the query efficiency is relatively high. Figure 8 shows the performance of two storage systems in both throughput and latency. As shown in Figure 8a, the QPS of CMSS reaches 688,188 and the QPS of HLF reaches 409,563 on average. CMSS has a higher QPS than HLF in any situation and CMSS improves the QPS by 0.68× on average. From the perspective of time consumption, due to the optimization of CMSS in reading and writing processes and system architecture, most of the data queried in this experiment will hit the cache of CMSS. As shown in Figure 8b, the time consumption of CMSS is significantly less than that of HLF, by almost 5 times.

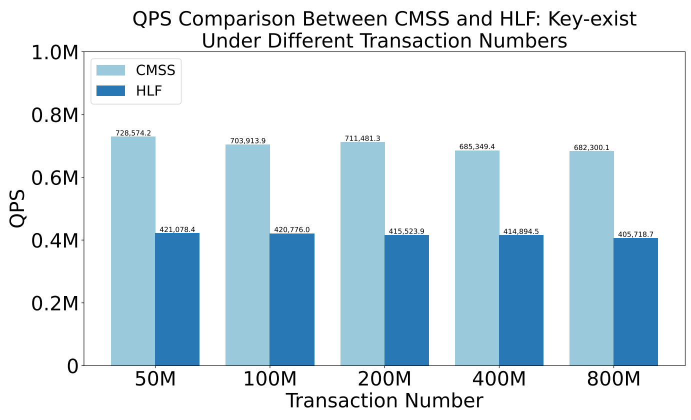


(**a**) QPS comparison in key-exist read last written experiment (**b**) Average time comparison in key-exist read last written experiment

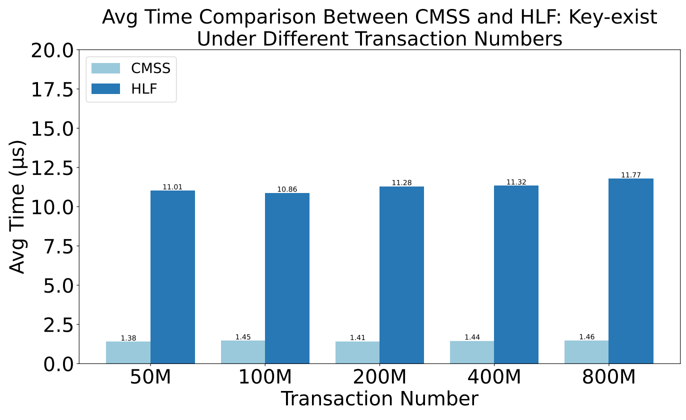
Figure 8. The experimental data from key-exist read last written comparison.

6.2.3. Key-Exist Random Read

Different from the previous experiment, this experiment uses random reading to query, which will make part of the query content unable to be hit in the cache, so that the overall read operation efficiency is low. In testbench, we use a random function to generate query content with a random block height. As depicted in Figure 9, our proposed CMSS achieves higher throughput and lower latency than HLF. Figure 9a shows that CMSS improves the QPS by 0.75× and the average time of HLF is almost 7.5 times the average time of CMSS.



(**a**) QPS comparison in key-exist random read experiment

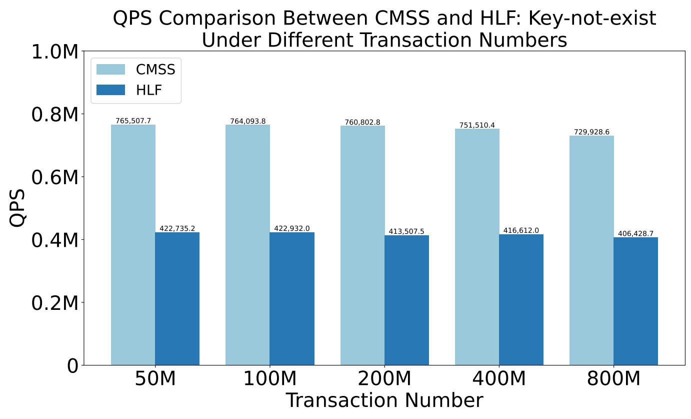


(**b**) Average time comparison in key-exist random read experiment

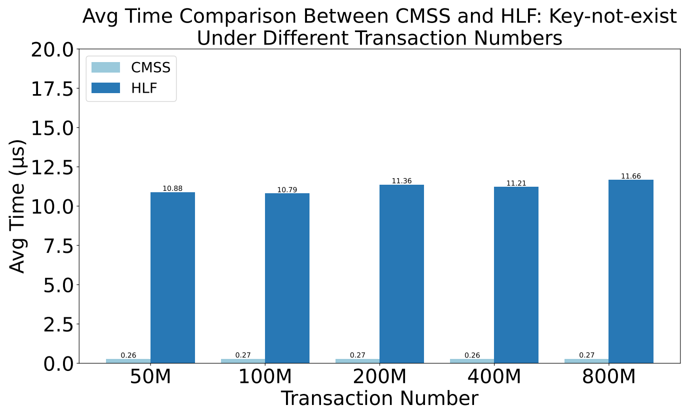
Figure 9. The experimental data from key-exist random read comparison.

6.2.4. Key-Not-Exist Read

In the blockchain system, transaction prevention is an important step that can keep the consistency and atomicity of blockchain. In transaction prevention, reading the key-not-exist content is one necessary component. We build this experiment to read the key-not-exist content in CMSS and HLF. Figure 10 shows that our proposed CMSS has better performance in both QPS and average time than HLF. As shown in Figure 10a, the QPS of CMSS is 1.83 times of the QPS of HLF on average. Under such throughput conditions, CMSS is still more efficient than HLF. Figure 10b shows that the average time of CMSS is just 0.025 times the average time of HLF.



(**a**) QPS comparison in key-not-exist experiment



(**b**) Average time comparison in key-not-exist experiment

Figure 10. The experimental data from key-not-exist comparison.

6.3. Storage Performance Comparison between CMSS and HLF

To evaluate the storage performance of our proposed CMSS, we build a storage experiment and check the storage status of disks. In this experiment, we conduct 100,000 blocks and each block has 10,000 transactions. In each transaction, there is one key–value pair and the size of the key and value are both 8 bits. Table 1 shows the storage performance of CMSS. The original data volume is 320 GB if we do not use hot–cold separation. When using hot–cold separation, the data that is not frequently accessed will be moved to cold storage and compressed. The data in Table 1 shows that CMSS has efficient performance in data storage and can significantly reduce data storage pressure.

Table 1. Storage performance of hot–cold separation.

Hot–Cold Separation Rate	Hot Disk Volume	Cold Disk Volume	Original Data Volume
50%	160 GB	32 GB	320 GB
90%	32 GB	57 GB	320 GB
99%	3.2 GB	63 GB	320 GB
99.9%	1 GB	64 GB	320 GB

To evaluate the economic cost savings of CMSS in a real production environment, we selected five well-known cloud service providers (three overseas providers: Microsoft Azure, Amazon Cloud and Google Cloud; two domestic providers: Tencent Cloud and Alibaba Cloud) to conduct economic cost testing.

As shown in Table 2, regardless of the pricing of any cloud service provider, CMSS can significantly reduce the storage cost of the entire system. When setting the hot–cold separation rate as 50%, CMSS can reduce economic costs by approximately half. It is worth noting that the price in the table is only the price for one month of service provided by the cloud service provider. In actual business, the service provision time often lasts for several months or even more than a year. From this point of view, CMSS can significantly reduce economic costs.

Table 2. Economic cost performance of hot–cold separation.

Cloud Service Provider	Hot Disk Unit Price *	Cold Disk Unit Price *	Hot–Cold Separation Rate	Total Storage Cost *
Tencent Cloud	1.7600 RMB/GB	0.0100 RMB/GB	None	563.20 RMB
			50%	281.92 RMB
			90%	56.89 RMB
			99%	6.26 RMB
			99.9%	2.40 RMB
Alibaba Cloud	1.00 RMB/GB	0.0075 RMB/GB	None	320.00 RMB
			50%	160.24 RMB
			90%	32.43 RMB
			99%	3.67 RMB
			99.9%	1.48 RMB
Microsoft Azure	2.2318 RMB/GB	0.0071 RMB/GB	None	714.18 RMB
			50%	357.32 RMB
			90%	71.82 RMB
			99%	7.59 RMB
			99.9%	2.69 RMB
Amazon Cloud	1.1519 RMB/GB	0.0071 RMB/GB	None	368.61 RMB
			50%	184.53 RMB
			90%	37.27 RMB
			99%	4.13 RMB
			99.9%	1.61 RMB
Google Cloud	1.4687 RMB/GB	0.0086 RMB/GB	None	469.98 RMB
			50%	235.27 RMB
			90%	47.49 RMB
			99%	5.24 RMB
			99.9%	2.02 RMB

* Here we only calculate one month of cloud service provision time. RMB stands for Renminbi, which is the official currency of the People's Republic of China. It is abbreviated as CNY internationally.

6.4. Experiment Results

In conclusion, a series of experiments were conducted to compare the performance of our proposed CMSS with Hyperledger Fabric (HLF). First, we evaluate the performance of reading and writing. The experiment in Section 6.2.1 demonstrates that CMSS outperforms HLF in terms of QPS and the average processing time of CMSS is shorter than HLF. These results indicate that CMSS exhibits superior performance over HLF in writing blocks, allowing it to process and store more blocks in the same time slot. Subsequent experiments in Sections 6.2.2–6.2.4 evaluate the reading performance. Different types of keys are used to represent various smart contracts in practical applications. The results show that CMSS outperforms HLF in reading blocks, resulting in a faster response time when querying information on the chain. Furthermore, the experiment in Section 6.3 compares CMSS at different hot–cold separation rates based on cloud storage services from five domestic and foreign cloud service providers. Through cost calculations in various settings, the experiments demonstrate that the hot–cold separation provided by CMSS can significantly reduce resource usage and economic costs. This makes it feasible for more participants to deploy full nodes to maintain the full data stored in the blockchain. In traditional techniques, this is not affordable for major participants, which may lead to reliability problems with data.

7. Conclusions

To solve the storage pressure and subsequent expensive costs caused by massive data, a blockchain storage system with high read–write performance and horizontal scaling support needs to be proposed urgently. In this paper, we propose **C**hain**M**aker **S**torage **S**ystem (CMSS). CMSS is a blockchain storage system with high read–write performance and horizontal scaling support. Our proposed CMSS provides a new block storage workflow that can improve the reading and writing efficiency while ensuring the consistency of the ledger. The **M**eta **F**ile **S**ystem (MFS) in CMSS supports horizontal expansion without reorganization and synchronization of ledger data. The hot–cold separation provided by CMSS greatly reduces resource usage and economic costs. In the experiments, we evaluate our CMSS with the most popular permissioned blockchain platform Hyperledger Fabric in read–write performance. We select five well-known cloud service providers to calculate the storage cost in the real production environment. The results show that our CMSS has higher throughput and lower latency than HLF in read and write operations and advantages in storage capacity and price.

In the future, we will focus on the blockchain consensus to optimize blockchain workflow and achieve higher efficiency and lower costs while ensuring security and privacy. What is more, we will try to add an intermediate layer in the architecture to achieve cross-platform compatibility.

Author Contributions: Methodology, W.Y.; Software, Y.C.; Writing—original draft, W.Y.; Writing—review & editing, C.L. and Y.C.; Visualization, M.G.; Funding acquisition, M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by National Key Research and Development Program of China under grant No. 2021YFB2701203, National Natural Science Foundation of China (NSFC) under grant No. 62172040, and Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: Author Wenjin Yang, Meng Ao and Yongqing Chen were employed by the company Tencent Inc. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Nakamoto, S.; Bitcoin, A. A Peer-to-Peer Electronic Cash System. 2008, Volume 4, p. 15. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 24 April 2024).
2. Storublevtcev, N. Cryptography in blockchain. In Proceedings of the Computational Science and Its Applications–ICCSA 2019: 19th International Conference, Saint Petersburg, Russia, 1–4 July 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 495–508.
3. Gao, Z.; Cao, L.; Du, X. Data Right Confirmation Mechanism Based on Blockchain and Locality Sensitive Hashing. In Proceedings of the 2020 3rd International Conference on Hot Information-Centric Networking (HotICN), Hefei, China, 12–14 December 2020; pp. 1–7. [CrossRef]

4. Wang, Q.; Li, R.; Wang, Q.; Chen, S. Non-fungible token (NFT): Overview, evaluation, opportunities and challenges. *arXiv* **2021**, arXiv:2105.07447.
5. Mukhopadhyay, U.; Skjellum, A.; Hambolu, O.; Oakley, J.; Yu, L.; Brooks, R. A brief survey of cryptocurrency systems. In Proceedings of the 2016 14th Annual Conference on Privacy, Security and Trust (PST), Auckland, New Zealand, 12–14 December 2016; pp. 745–752.
6. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
7. Etherscan. Ethereum Full Node Sync Archive Chart. 2024. Available online: https://etherscan.io/ (accessed on 24 April 2024).
8. Ycharts: Bitcoin Blockchain Size. 2024. Available online: https://ycharts.com/indicators/bitcoin_blockchain_size (accessed on 24 April 2024).
9. Zamani, M.; Movahedi, M.; Raykova, M. Rapidchain: Scaling blockchain via full sharding. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 931–948.
10. Dang, H.; Dinh, T.T.A.; Loghin, D.; Chang, E.C.; Lin, Q.; Ooi, B.C. Towards scaling blockchain systems via sharding. In Proceedings of the 2019 International Conference on Management of Data, Amsterdam, The Netherlands, 30 June–5 July 2019; pp. 123–140.
11. Qi, X. S-store: A scalable data store towards permissioned blockchain sharding. In Proceedings of the IEEE INFOCOM 2022—IEEE Conference on Computer Communications, London, UK, 2–5 May 2022; pp. 1978–1987.
12. Gaži, P.; Kiayias, A.; Zindros, D. Proof-of-stake sidechains. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019; pp. 139–156.
13. Yin, L.; Xu, J.; Tang, Q. Sidechains with fast cross-chain transfers. *IEEE Trans. Dependable Secur. Comput.* **2021**, *19*, 3925–3940. [CrossRef]
14. Chainmaker. Available online: https://chainmaker.org.cn/home (accessed on 8 April 2024).
15. Luu, L.; Narayanan, V.; Zheng, C.; Baweja, K.; Gilbert, S.; Saxena, P. A secure sharding protocol for open blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 17–30.
16. Wang, J.; Wang, H. Monoxide: Scale out blockchains with asynchronous consensus zones. In Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19), Boston, MA, USA, 26–28 February 2019; pp. 95–112.
17. Al-Bassam, M.; Sonnino, A.; Bano, S.; Hrycyszyn, D.; Danezis, G. Chainspace: A sharded smart contracts platform. *arXiv* **2017**, arXiv:1708.03778.
18. Hellings, J.; Sadoghi, M. Byshard: Sharding in a byzantine environment. *VLDB J.* **2023**, *32*, 1343–1367. [CrossRef]
19. Hong, Z.; Guo, S.; Zhou, E.; Chen, W.; Huang, H.; Zomaya, A. Gridb: Scaling blockchain database via sharding and off-chain cross-shard mechanism. *Proc. Vldb Endow.* **2023**, *16*, 1685–1698. [CrossRef]
20. Back, A.; Corallo, M.; Dashjr, L.; Friedenbach, M.; Maxwell, G.; Miller, A.; Poelstra, A.; Timón, J.; Wuille, P. Enabling Blockchain Innovations with Pegged Sidechains. 2014, Volume 72, pp. 201–224. Available online: http://kevinriggen.com/files/sidechains.pdf (accessed on 24 April 2024).
21. Lerner, S. Drivechains, Sidechains and Hybrid 2-way Peg Designs. Available online: https://docs.rsk.co/Drivechains_Sidechains_and_Hybrid_2-way_peg_Designs_R9.pdf (accessed on 24 April 2024).
22. Kiayias, A.; Lamprou, N.; Stouka, A.P. Proofs of proofs of work with sublinear complexity. In Proceedings of the Financial Cryptography and Data Security: FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, 26 February 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 61–78.
23. Bünz, B.; Kiffer, L.; Luu, L.; Zamani, M. Flyclient: Super-light clients for cryptocurrencies. In Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 18–21 May 2020; pp. 928–946.
24. Yin, L.; Xu, J.; Liang, K.; Zhang, Z. Sidechains with optimally succinct proof. *IEEE Trans. Dependable Secur. Comput.* **2023**, 1–15. [CrossRef]
25. Deng, Z.; Li, T.; Tang, C.; He, D.; Zheng, Z. PSSC: Practical and Secure Sidechains Construction for Heterogeneous Blockchains Orienting IoT. *IEEE Internet Things J.* **2023**, *11*, 4600–4613. [CrossRef]
26. Qi, X.; Zhang, Z.; Jin, C.; Zhou, A. BFT-Store: Storage partition for permissioned blockchain via erasure coding. In Proceedings of the 2020 IEEE 36th International Conference on Data Engineering (ICDE), Dallas, TX, USA, 20–24 April 2020; pp. 1926–1929.
27. Du, Z.; Qian, H.f.; Pang, X. Partitionchain: A scalable and reliable data storage strategy for permissioned blockchain. *IEEE Trans. Knowl. Data Eng.* **2021**, *35*, 4124–4136. [CrossRef]
28. Bagozi, A.; Bianchini, D.; De Antonellis, V.; Garda, M.; Melchiori, M. A three-layered approach for designing smart contracts in collaborative processes. In Proceedings of the On the Move to Meaningful Internet Systems: OTM 2019 Conferences: Confederated International Conferences: CoopIS, ODBASE, C&TC 2019, Rhodes, Greece, 21–25 October 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 440–457.
29. Bagozi, A.; Bianchini, D.; De Antonellis, V.; Garda, M.; Melchiori, M. A blockchain-based approach for trust management in collaborative business processes. In Proceedings of the Web Information Systems Engineering–WISE 2021: 22nd International Conference on Web Information Systems Engineering, WISE 2021, Melbourne, VIC, Australia, 26–29 October 2021; Springer: Berlin/Heidelberg, Germany, 2021; pp. 59–67.

30. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference, Porto, Portugal, 23–26 April 2018; pp. 1–15.
31. Goquorum. 2022. Available online: https://github.com/ConsenSys/quorum (accessed on 24 April 2024).
32. Preneel, B. Cryptographic hash functions. *Eur. Trans. Telecommun.* **1994**, *5*, 431–448. [CrossRef]
33. Ghemawat, S.; Dean, J. LevelDB. 2011. Available online: https://leveljs.org/ (accessed on 24 April 2024).
34. Anderson, J.C.; Lehnardt, J.; Slater, N. *CouchDB: The Definitive Guide: Time to Relax*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2010.
35. Chodorow, K. *Scaling MongoDB: Sharding, Cluster Setup, and Administration*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2011.
36. Platt, M.; McBurney, P. Sybil in the haystack: A comprehensive review of blockchain consensus mechanisms in search of strong Sybil attack resistance. *Algorithms* **2023**, *16*, 34. [CrossRef]
37. Zhang, Z.; Liu, X.; Li, M.; Yin, H.; Zhu, L.; Khoussainov, B.; Gai, K. HCA: Hashchain-based Consensus Acceleration via Re-voting. *IEEE Trans. Dependable Secur. Comput.* **2023**, *21*, 775–788. [CrossRef]
38. Zhang, Z.; Feng, K.; Chen, X.; Liu, X.; Sun, H. RHCA: Robust HCA via Consistent Revoting. *Mathematics* **2024**, *12*, 593. [CrossRef]
39. Liu, J.; Li, P.; Cheng, R.; Asokan, N.; Song, D. Parallel and asynchronous smart contract execution. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *33*, 1097–1108. [CrossRef]
40. Cheng, J.C.; Lee, N.Y.; Chi, C.; Chen, Y.H. Blockchain and smart contract for digital certificate. In Proceedings of the 2018 IEEE International Conference on Applied System Invention (ICASI), Chiba, Japan, 13–17 April 2018; pp. 1046–1051.
41. Hao, J.; Gao, J.; Xiang, P.; Zhang, J.; Chen, Z.; Hu, H.; Chen, Z. TDID: Transparent and Efficient Decentralized Identity Management with Blockchain. In Proceedings of the 2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Oahu, HI, USA, 1–4 October 2023; pp. 1752–1759.