Poster Abstract: OrderlessFile: A CRDT-Enabled Permissioned Blockchain for File Storage

Pezhman Nasirifard
Technical University of Munich
Munich, Germany
p.nasirifard@tum.de

Ruben Mayer
Technical University of Munich
Munich, Germany
ruben.mayer@tum.de

Hans-Arno Jacobsen
University of Toronto
Toronto, Canada
jacobsen@eecg.toronto.edu

ABSTRACT

Cloud storage has gained popularity as an affordable and available cloud-based file storage. However, despite its apparent advantages, clients must rely on cloud providers to safely and securely store the files. A few blockchain-based file storage systems have been introduced as trusted alternatives. However, their limited scalability and use of off-chain storage restrict their applicability. To address these issues, we introduce OrderlessFile, a CRDT-based on-chain permissioned blockchain for file storage.

CCS CONCEPTS

• **Computer systems organization → Distributed architectures**; • **Information systems → Information storage systems**.

KEYWORDS

Blockchain, Conflict-free Replicated Data Type, File Storage

ACM Reference Format:
Pezhman Nasirifard, Ruben Mayer, and Hans-Arno Jacobsen. 2022. Poster Abstract: OrderlessFile: A CRDT-Enabled Permissioned Blockchain for File Storage. In *23rd International Middleware Conference Demos and Posters (Middleware '22), November 7–11, 2022, Quebec, QC, Canada.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3565386.3565491

1 INTRODUCTION

In recent years cloud storage services have become an integral part of cloud ecosystems [2]. Although cloud storage offers affordable and available Storage-as-a-Service, most cloud providers lack transparency on the privacy and security of stored files [14]. Clients must trust the service-level agreements and be confident that providers do not tamper with data and store them safely. Several blockchain-based distributed and decentralized file storage systems have been proposed to address these issues [2, 14]. Although blockchain-based file storage systems are more secure and private than conventional cloud providers, several existing systems are based on *Proof-of-Work-based (PoW)* solutions, such as Ethereum [5, 13, 14] and suffer from the common scalability issues of PoW-based protocols. Furthermore, due to the storage limitations on blockchains, exiting solutions use various off-chain systems such as *InterPlanetary File System (IPFS)*, which might present other security and privacy issues.

To address the scalability issues and limitations of stored file size, we introduce OrderlessFile, a private and distributed permissioned blockchain-based file storage. OrderlessFile offers a scalable and safe protocol for replicating and storing encrypted files where Byzantine participants cannot tamper with data and violate its integrity. We also introduce FileCRDT, a customized *Conflict-free Replicated Data Type (CRDT)* [10] for enabling the clients to split files into shards that are stored and replicated on OrderlessFile. Like conventional file storage, OrderlessFile can store various types of data from IoT to extensive electric vehicles, and machine learning datasets [2, 4, 9, 14].

2 FILECRDT: A CRDT FOR FILE STORAGE

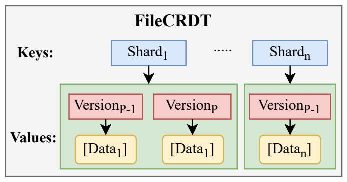

Figure 1: Internal structure of FileCRDT.

OrderlessFile uses FileCRDT for sharding and storing files. The structure of FileCRDT is shown in Figure 1. Each file stored on OrderlessFile has a unique identifier and is versioned based on a logical clock tracked by the client. The logical clock indicates the file's version number, and the client increments it with every new file version. FileCRDT is a nested data structure in the format of key-value pairs. Depending on the file size and the OrderlessFile's configuration settings, the file is split into shards, where each shard is stored as a key-value pair in an instance of FileCRDT. The key is the shard's identifier which is unique per file. The value is another key-value data structure, where the key is the file's version, and the value is a multi-value register [3] containing the shard data. The purpose of keeping track of the versions is to resolve the conflicting updates sent by the client and prevent data corruption as the shards are gradually transmitted to be stored.

3 ARCHITECTURE AND PROTOCOL

OrderlessFile consists of several organizations and clients. The organizations are responsible for receiving and storing files from clients. The clients can communicate with every non-failed organization and upload and download files. For uploading files, clients follow a two-phase *endorsement-storage* protocol, as shown in Figure 2.

Endorsement Phase – Depending on the file size and the maximum allowed shard size, the client first splits the file into shards. The allowed shard size is a global system configuration setting. For every shard, a shard signature $SS_m = Hash(<Encrypt(Shard_m), Version_p>)$ is created based on the shard's encrypted data and the file's current version. SS_m is sent to the organizations (Step 1 in Figure 2). The

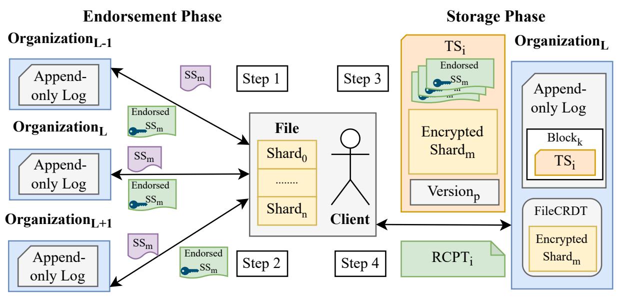

Figure 2: Workflow for uploading files.

client sends SS_m to all organizations or a subset of them, depending on the replication factor intended for storing the file. The organization signs SS_m with its private key based on public-key cryptography and sends the response (a.k.a endorsements), to the client (Step 2).

Storage Phase – The client waits to receive the endorsements from organizations and verifies their validity by using the organization's public key. If every endorsement is valid, the client creates a transaction TS_i that contains the endorsements, the encrypted shard data, the shard's identifier, and the file's version. The client sends the transaction to the same organizations (Step 3). The organization appends the transaction to an append-only hash-chain log. The organization first creates a block $Block_k = < Hash(TS_i), Hash(Block_{k-1}) >$, which contains the hash value of TS_i and the hash of the previous block and appends the block to the log. Afterward, the organization validates every endorsement in TS_i to ensure that the organization endorsed the identical SS_m. If the transaction is valid, a receipt $RCPT_i = HashAndSign(Block_k, Valid)$, is created and sent to the client (Step 4). For an invalid transaction, the organization sends a rejection. Since the created block contains the hash of the transaction and the previous block, the Byzantine organization cannot tamper with the content of the transaction without invalidating the receipt of TS_i and the previous transactions. If TS_i is valid, the organization updates the instance of FileCRDT that contains the file. The shard in FileCRDT is modified based on the shard's identifier and the file's version. If a conflicting shard with an identical version exists, the shard data is added to the multi-value register, and the client decides which data to use. Finally, the organization iterates through the shards to verify whether every shard has a $Version_p$ key. If $Version_p$ key exits in every shard in the FileCRDT instance, the key-value pairs with the key $Version_{p-1}$ are removed to free up disk space.

For downloading a file, the client sends a request to an organization that stores the file with the file identifier and receives the shards with the latest version.

We open-sourced the code [1] and published an extended paper on the permissioned blockchain part of the system [7, 8].

4 EVALUATION

For evaluating OrderlessFile, we deployed a network of 16 organizations with 1000 clients downloading and uploading files concurrently. The workload consists of half download and half upload transactions uniformly distributed during the experiments. Each client owns one file consisting of ten shards with a replication factor of two. We gradually increased the maximum shard size from 25KB to 100KB. The average latency to the throughput of upload and download transactions is shown in Figure 3. For uploading transactions, latency remains constant for smaller shard sizes. However, the latency increases for larger shard sizes as they result in larger transactions requiring more time to be transmitted, and the overhead for writing larger shards to the disk increases. The latency for download requests increases as throughput increases since the load on the bandwidth increases.

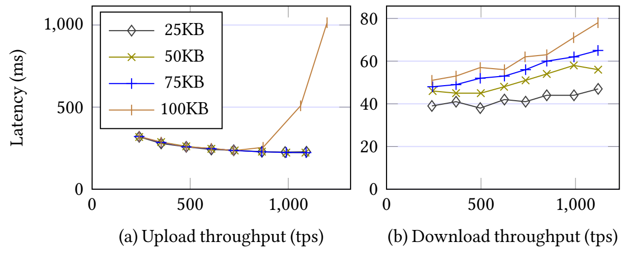

Figure 3: Latency to throughput for an increasing shard size.

5 RELATED WORK

Some studies proposed CRDT-based decentralized file storage in non-Byzantine distributed environments [11, 12]. However, the applicability of CRDT-enabled blockchains for file storage has received limited attention [6]. A few works proposed solutions for decentralized blockchain-based file storage [1, 5, 13]. Stroj [5] is an Ethereum-based cloud storage where providers can rent out their excess hardware and bandwidth to the clients. Sia [13] also offers a PoW-based solution for providers to rent out their excess hardware. FileCoin [1] provides blockchain-based off-chain storage using IPFS.

REFERENCES

[1] J. Benet and N. Greco. 2018. Filecoin: A Decentralized Storage Network. (2018).
[2] N. Deepa, Q.-V. Pham, D. C. Nguyen, and *et al.* 2022. A Survey on Blockchain for Big Data: Approaches, Opportunities, and Future Directions. *FGCS* (2022).
[3] M. Kleppmann and A. R. Beresford. 2017. A Conflict-free Replicated JSON Datatype. *IEEE TPDS* (2017).
[4] F. Kohlbrenner, P. Nasirifard, C. Löbel, and H.-A. Jacobsen. 2019. A Blockchain-Based Payment and Validity Check System for Vehicle Services. In *Middleware Demos and Posters*.
[5] Storj Labs. 2018. Storj: A Decentralized Cloud Storage Network Framework.
[6] P. Nasirifard, R. Mayer, and H.-A. Jacobsen. 2019. FabricCRDT: A Conflict-free Replicated Datatypes Approach to Permissioned Blockchains. In *Middleware*.
[7] P. Nasirifard, R. Mayer, and H.-A. Jacobsen. 2022. OrderlessChain: A CRDT-Enabled Blockchain Without Total Global Order of Transactions. In *Middleware Demos and Posters*.
[8] P. Nasirifard, R. Mayer, and H.-A. Jacobsen. 2022. OrderlessChain: Do Permissioned Blockchains Need Total Global Order of Transactions? *CoRR* (2022). https://doi.org/10.48550/arXiv.2210.01477 arXiv:2210.01477
[9] P. Nasirifard, R. Mayer, and H.-A. Jacobsen. 2022. OrderlessFL: A CRDT-Enabled Permissioned Blockchain for Federated Learning. In *Middleware Demos and Posters*.
[10] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski. 2011. Conflict-free Replicated Data Types. In *SSS*.
[11] V. Tao, M. Shapiro, and V. Rancurel. 2015. Merging Semantics for Conflict Updates in Geo-distributed File Systems. In *ACM SYSTOR*.
[12] R. Vaillant, D. Vasilas, M. Shapiro, and T. L. Nguyen. 2021. CRDTs for Truly Concurrent File Systems. In *ACM HotStorage Workshop*.
[13] D. Vorick and L. Champine. 2014. Sia: Simple Decentralized Storage. (2014).
[14] N. Zahed Benisi, M. Aminian, and B. Javadi. 2020. Blockchain-based Decentralized Storage Networks: A Survey. *J. Netw. Comput. Appl.* (2020).

[1] https://github.com/orderlesschain/orderlessfile