A Scalable Storage Scheme for On-Chain Big Data Using Historical Blockchains

Marcos Felipe and Haiping Xu*
Computer and Information Science Department
University of Massachusetts Dartmouth
Dartmouth, MA, 02747, USA
mfelipe1@umassd.edu, hxu@umassd.edu
**corresponding author*

***Abstract*—Despite the growing interest in blockchain technology, the scalability of blockchain storage has become a major issue for applications that require large amounts of on-chain data. In this paper, we propose a novel scalable storage scheme for consortium networks to manage the storage capacity required by data-rich blockchain applications. We establish network nodes as super peers or regular peers, where super peers can maintain old blockchain data in the form of historical blockchains. Regular peers maintain only the latest blockchain data stored in the current blockchain, but they can access any data in the historical blockchains by making queries to the super peers. We present procedures to build a historical blockchain and retrieve data from the historical blockchains and the current blockchain in a concurrent manner. Experimental results show that our scalable storage scheme using historical blockchains is feasible and effective in accessing and sharing healthcare data with image files.**

Keywords-consortium blockchain; historical blockchain; on-chain data; scalable storage; super peer; healthcare data

I. INTRODUCTION

Blockchain technology is becoming very popular today as a decentralized distributed ledger used for securing data and transactions while providing users with a reliable and convenient way to access data. The technology allows peer-to-peer networks to maintain "chains" of blocks that contain transaction records. Each block stores a hash value of its previous block, so that any attempted modification to a block affects all subsequent blocks in the chain. In addition, peers in the network maintain copies of the chain, so transactions and blocks can be verified. In a permissioned blockchain, all peers with the required permissions have access to the transactions recorded in the network, but the transaction records are secure and immutable, ensuring data integrity and transparency.

The convenience offered by blockchain as a form of data storage has led to a rise in popularity over the past decade. Bitcoin, a digital currency that uses public blockchain technology, has over 100 million users in 2022. From January 2012 to July 2022, the Bitcoin blockchain has grown by more than 400 gigabytes and has even doubled since February 2019. If this massive growth continues, the cost of being a full node in a blockchain network will become completely unrealistic for regular peers. Storage issues also pose problems for non-public blockchain networks, such as private and consortium networks. While such networks do not have as many peers involved in as many transactions, storage issues can still arise, depending on the type of data being stored. Blockchain technology has been used in many different areas, including healthcare, real estate, insurance, and the Internet of Things (IoT). These types of applications often use consortium blockchain networks, but because they can be very data-rich, scalability of blockchain storage has been a major concern.

In recent years, there have been many studies on consortium blockchain storage management [1][2]. However, most of these efforts involve the use of off-chain storage, employing solutions such as InterPlanetary File System (IPFS) or cloud storage. By using these methods, the storage problem can be mitigated, but since these methods store most of the data off-chain, the benefits of using blockchain technology to secure and maintain the data are lost. As the major contribution of this paper, we propose an on-chain method that reduces the storage burden on the majority of peers in a network by splitting a current blockchain (CB) into a historical blockchain (HB) and a new CB. An HB is an immutable blockchain that contains historical data of the blockchain, while a CB contains blockchain data from the last few years, which can grow until it needs to be split again. We establish the network nodes as either super peers or regular peers, with a reasonable number of super peers maintaining CB and multiple HBs. Most nodes are regular peers, maintaining only CB, but they can access any data in HBs by making queries to the super peers. Using a time-based partitioning method, when the CB reaches a certain age, say 10 years, it can be split into an HB and a new CB, containing data from the earlier 5 years and the last 5 years, respectively. The group of super peers is responsible for maintaining the HBs and the latest CB, while the regular peers store only the CB, allowing for a greatly reduced storage burden. As the CB continues to grow, its size can be reduced again and more HBs can be created and maintained by the super peers.

Although not required, regular peers can still store HBs; however, they are not responsible for handling queries to retrieve historical data made by other regular peers. When a regular peer maintains only the CB, it can access the HBs by querying a super peer in the network. Once the super peer retrieves the requested data from the HBs, it notifies the requesting regular peer and allows it to download the data. To ensure that retrieving historical data remains an efficient task for fairly large blockchains with many years of data, we also introduce a meta-block for the CB and each HB. The meta-block contains an index of user transaction records stored in the CB or an HB to speed up the data retrieval process.

2693-9371/22/$31.00 ©2022 IEEE
DOI 10.1109/QRS-C57518.2022.00017

II. Related Work

The issue of scalability in blockchain applications has been an ongoing concern, especially in public blockchain applications. Poon and Dryja introduced the Bitcoin Lightning Network, which is a decentralized system where transactions are sent through channels for off-chain value transfer [3]. The use of the Bitcoin Lightning Network, which includes micro-payments sent continuously between two parties, significantly reduces global Bitcoin blockchain transaction broadcasts and makes the Bitcoin network scalable. Further research on blockchain scalability facilitated by off-chain strategies is more applicable to private and consortium blockchain applications. To reduce the high computation and storage costs in blockchain-based applications, Eberhardt and Tai investigated different off-chain computation and storage methods [4]. They presented five off-chain patterns for moving computation and data off the blockchain without violating the trustless property. Wang et al. proposed ChainSplitter, an off-chain scalability solution for Industrial Internet of Things (IIoT) blockchain applications [5]. The proposed approach featured a hierarchical storage structure that stores recent blocks in an overlay network, with most of the blockchain data stored in the cloud. Although the cloud is organized as a distributed cloud storage, the blockchain data in the cloud is not maintained by peers; therefore, the cloud acts as an off-chain storage for the blockchain data. IPFS is a decentralized, verifiable, blockchain-compatible distributed storage system. IPFS used with blockchain networks as an off-chain approach also provides a scalable solution for blockchains. Li et al. were able to reduce the asset size of transactions and increase the transaction throughput of an experimental consortium blockchain network by storing hash values of encrypted data on-chain and storing the encrypted data itself off-chain in an associated IPFS [6]. They provided a solution for secure storage and access to a task-scheduling scheme by integrating Hyperledger Fabric with IPFS services. Although the off-chain approaches provide feasible ways to mitigate the scalability issue of blockchains, as noted in [4], the fundamental properties of blockchains and blockchain applications can be compromised to varying degrees when using off-chain approaches. In contrast, our approach stores the big data in historical blockchains and does not use off-chain storage; thus, all the essential blockchain properties of the stored data are maintained in our proposed approach.

Attempts to use on-chain storage to address the scalability issue are very rare. Xu proposed the section-blockchain, an on-chain method to reduce storage cost of blockchain networks for devices with insufficient storage [7]. In their approach, instead of implementing lightweight nodes, all nodes store parts of the complete blockchain equally and are incentivized to change their local storage to receive more payoffs. In a further attempt, segmented blockchains were proposed to enable nodes to store a copy of one blockchain segment [8]. They showed that their approach can reduce the storage cost of a blockchain system while maintaining decentralization without compromising the security of the blockchain. Thamrin and Xu proposed a framework for cloud-based blockchains to store multimedia files securely and reliably [9]. They used the cloud-based blockchain as a complete blockchain for data accessibility, redundancy, and security, while a lite blockchain allows local storage of text-based information and metadata for multimedia files. Although the above methods allow big data storage, data retrieval can be inefficient due to the incomplete data stored in some blockchains. Unlike them, we divide a complete blockchain into a current blockchain and multiple historical blockchains, maintained by super peers. A regular peer can access its local current blockchain and request historical blockchain data from a super peer in a concurrent manner.

Blockchain technology has been extensively used in healthcare systems. Jayabalan and Jeyanthi introduced a blockchain-based application using IPFS specifically for healthcare systems [10]. Focusing on storage of electronic health records (EHRs), the IPFS service was used to move data off-chain while retaining hashes of the data on the blockchain. Im et al. proposed a consortium blockchain for patient access and management of personal health records (PHRs), implemented using Hyperledger Fabric [11]. By comparing with the public blockchain Ethereum, they concluded that Hyperledger Fabric was a viable approach to ensure the privacy of PHRs. Recently, Thamrin and Xu proposed a hierarchical cloud-based consortium blockchain for the storage of EHRs [12]. In their approach, big data such as multimedia files can be stored in a cloud-based hospital blockchain network within a local area and shared with hospitals outside the networks through high-level blockchain networks, called city and state blockchain networks. Due to the hierarchical structure of the blockchain networks, which supports concurrent search and retrieval of EHRs, this could be an efficient way to access and share EHRs nationwide. Although the above approaches provide feasible ways for the application of blockchain technology in healthcare systems, the scalability issue remains a major concern. In our approach, since the current blockchain and each historical blockchain contain only a certain number of years of blockchain data, each blockchain is easier to manage and the blockchain scalability problem can be significantly alleviated.

III. Scalable Storage Using Historical Blockchains

A. A Framework for Scalable Blockchain Networks

Data storage technology has long been studied and improved, whether it is physical storage, cloud storage, or now blockchain networks. Due to the many benefits of using blockchain networks, including decentralization, security, immutability, transparency and traceability, blockchain technology has been widely used in many areas beyond cryptocurrency. Many blockchain-enabled applications require the storage of large amounts of data; therefore, managing the size of the blockchain is critical to maintaining its viability for nodes and networks. For example, in the healthcare domain, a single patient visit to a hospital can result in a great deal of data, and when that data scales to multiple visits, the storage load can become quite large. For a hospital with many patients, and then expanding to a local area or city network of hospitals that share information, the data problem only proliferates. If a local or a city network of hospitals is

going to consider using a consortium blockchain for data storage and sharing, they must address this scalability issue. Techniques that store medical data separately from the blockchain do provide viable solutions, but the benefits of using blockchain storage are compromised for any data that is stored off the blockchain. In this paper, we propose a method to maintain medical data on-chain with the burden shouldered by a smaller group of well-equipped super peers, representing large hospitals within a local area. In the group of large hospitals, each hospital can dedicate the necessary resources to maintain older on-chain data in the form of historical blockchains, relieving the rest of the network from the burden of this data while retaining the benefits and convenience provided by the blockchain technology. Figure 1 shows the framework for a scalable blockchain network.

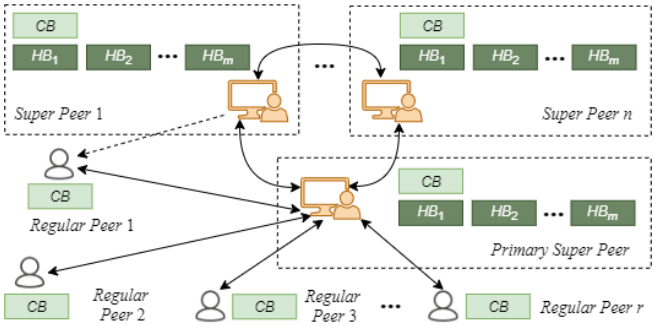

Figure 1. A framework for a scalable blockchain network

As shown in the figure, there are n+1 super peers within a blockchain network who are responsible for maintaining the current blockchain *CB* and all historical blockchains *HB*s. Super peers are also responsible for creating a new block and using a consensus process to verify and approve the new block, which makes regular peers highly lightweight. We define a *Primary Super Peer*, or *PSP*, an elected super peer who plays a role in efficiently facilitating access to data in *HB*s by regular peers. Regular peers maintain only the *CB* and can make queries to the *PSP* to retrieve historical data stored in *HB*s. The *PSP* can assign a query from a regular peer to a super peer, and the super peer can return the retrieved historical data directly to the requesting regular peer. More importantly, as described in Section III.D, when a current block reaches a certain age, a super peer can split it into a historical blockchain and a reduced current blockchain.

B. Block and Transaction Structures

The structure of a block is the foundation of the blockchain and is an integral part of its functionality. As shown in Fig. 2, a block contains a block header, which is defined as a 4-tuple (*B*, *T*, *S*, *H*), where *B* is the block ID, *T* is the timestamp when the block is created, *S* is the size of the list of transactions recorded in the block, and *H* is the hash value of the previous block. A block also contains a list of transactions, which are defined as a 4-tuple (*TI*, *TS*, *PI*, *TD*), where *TI* is the transaction ID, *TS* is the timestamp when the transaction is created, *PI* is the patient ID, and *TD* is the transaction data, including text-based information and images files. In addition, a block with block ID *bID* contains a list of digital signatures, $ds[bID]_v$, where *v* is a super peer who approves it as a new block in the consensus process. When a block has been approved as a new block by the majority of super peers, the hash value of the block is calculated by applying a hash function to the block file containing all the aforementioned components, and the hash value *hash*(*cur-Block*) is attached to the end of the block file.

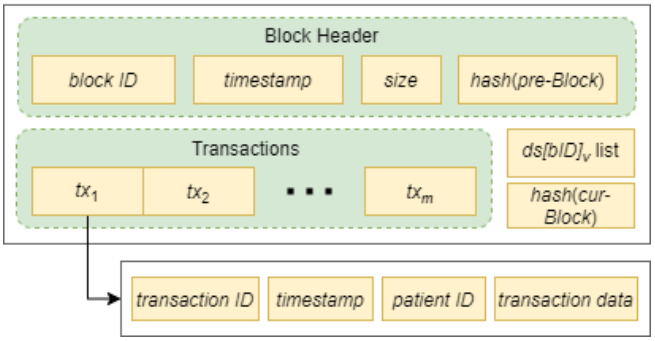

Figure 2. The structure of a block and a transaction

Note that to limit the block size, each block contains no more than 500 transactions and only contains transactions created during the same day. Thus, the last block created at the end of a day may contain less than 500 transactions.

C. The Structure of a Meta-Block

To support efficient data retrieval in blockchains, a *meta-block* is defined as a special block that stores metadata for each historical blockchain or the current blockchain. The meta-block is the only mutable block that is attached to the beginning of a blockchain. As shown in Fig. 3, a meta-block is defined as a 5-tuple (*SD*, *ED*, *SB*, *EB*, *HM*), where *SD* is the timestamp of the first transaction in the first block of the blockchain; *ED* is the timestamp of the last transaction in the last block of the blockchain; *SB* and *EB* are the block IDs of the first block and the last block in the blockchain, respectively; and *HM* is a HashMap containing a list of <*key*, *value*> pairs, where the key is a patient ID and the value is a list of locations that store transactions of the patient. Each location is defined as a triple (*B*, *A*, *O*), where *B* is the block ID, *A* is the address of the transaction in the block, and *O* is the offset of the transaction size.

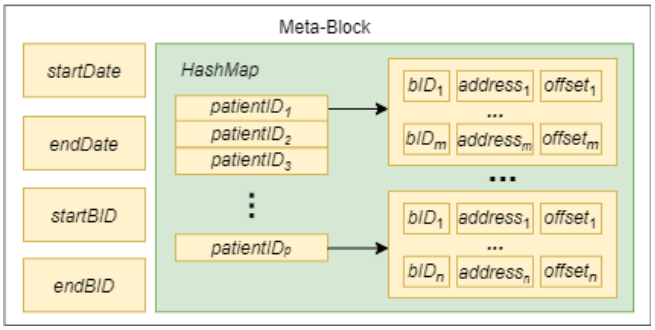

Figure 3. The structure of a meta-block

Note that the integrity of the metadata in a meta-block can be trusted, as the information can be reviewed, validated and

refreshed at any point in time by reading data from the relevant portion of the blockchain. The introduction of the meta-block reduces the search space and, therefore, it minimizes the search time for any query against the historical blockchains. With this metadata, queries for information on historical blockchains can be accomplished by finding the appropriate historical blockchain containing the information and extracting the relevant data from the blocks involved.

D. Generation of a Historical Blockchain

Initially, the current blockchain is the only blockchain. When a certain number of years is reached, say 10 years, the blockchain is split, with the last 5 years of data remaining in the current blockchain and the previous 5 years of data stored in a new historical blockchain. A new meta-block, containing the metadata of the historical blockchain, is attached to the beginning of the blockchain, and the meta-block of the current blockchain is refreshed. This process can be repeated when the current blockchain again contains 10 years of data.

Figure 4 shows how the current blockchain *CB* is split into a historical blockchain and a new current blockchain. Let the block IDs of the first and last block in *CB* be *m* and *n*, respectively. Note that $m = 1$ if the current blockchain has never been split before. Let block *k* be the most recent block in *CB*, which is at least 6 years old. We establish blocks *m* through *k* as a historical blockchain *HB* and generate a new meta-block MB_{HB} for it. Blocks $k+1$ through *n* persist as the updated current blockchain, while blocks *m* through *k* are deleted. The meta-block MB_{CB} associated with the current blockchain is refreshed by scanning the data in the new current blockchain (i.e., blocks $k+1$ through *n*). We now have an updated current blockchain and a historical blockchain, each containing 5 years of data.

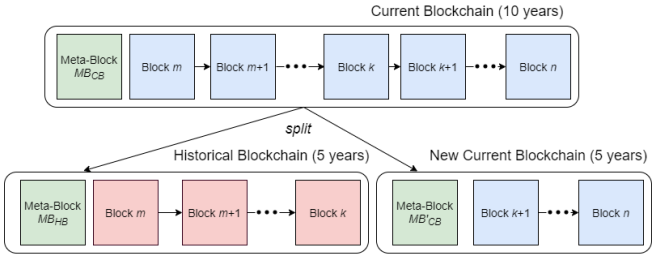

Figure 4. A blockchain split into a historical and a current blockchain

Algorithm 1 shows the process of splitting the current blockchain into a historical blockchain and an updated current blockchain. As shown in the algorithm, the meta-block of *HB* MB_{HB} contains the date of the first transaction in the first block of *HB*, the date of the last transaction in the last block of *HB*, and the block IDs of the first and the last blocks in *HB*. To create a HashMap that contains all <*key*, *value*> pairs, each block in *HB* is scanned, and each triple (*B*, *A*, *O*) associated with patient ID α is added to a list LT_α. Once the scanning process is completed, all pairs of $<\alpha, LT_\alpha>$ are added to the HashMap in MB_{HB}. Now in *CB*, all blocks that have been recorded in *HB* are removed, and the meta-block of the updated *CB* must be refreshed by removing all triples that reference transactions that have been transferred to *HB*. Finally, the new *HB* and the updated *CB* are returned.

Algorithm 1: Splitting a Current Blockchain

Input: A current blockchain *CB* containing 10 years of data
Output: Historical blockchain *HB* with 5 years of earlier data and an updated *CB* with the last 5 years of data

1. Let *m* and *n* be the IDs of the first and the last block in *CB*
2. Let *k* be the most recent block at least 6 years old, where $n > k$
3. Extract blocks *m* through *k* from *CB* and create a new historical blockchain *HB* with the k-m+1 blocks
4. Create an empty meta-block MB_{HB} associated with *HB*
5. Set *SD* in MB_{HB} as the date of the first transaction in block *m*
6. Set *ED* in MB_{HB} as the date of the last transaction in block *n*
7. Set *SB* and *EB* in MB_{HB} to *m* and *k*, respectively
8. **for** each block β in *HB*
9. Scan block β and add each triple (*B*, *A*, *O*) associated with *patientID* α to a list LT_α
10. Create a HashMap in MB_{HB} and add all pairs of $<\alpha, LT_\alpha>$ to it
11. Attach MB_{HB} to the beginning of *HB*
12. Remove blocks *m* through *k* from *CB*
13. Update *CB*'s meta-block MB_{CB} accordingly, as with MB_{HB}
14. **return** *HB* and *CB*

IV. Retrieval of Historical Blockchain Data

A. Load Balancing Data Retrieval Requests

Assume a regular peer queries the past 5*x* years of data from the blockchains, where $x \in [1, 5]$. When *x* equals 1, the regular peer can search patient information directly from its local blockchain, which must be at least 5 years old. When *x* is equal to 2 or more, it must make a query to the *PSP* to search for data from the historical blockchains. The request for such a query involves a patient ID (for which data is collected) and the number of years of data being search (i.e., the search length). As shown in Fig. 5, when the *PSP* receives a query from a regular peer, it acts as a director, balancing the load of the queries, and distributing them evenly based on the weights of queries fulfilled by super peers. Each query receives a weight that estimates the time to complete it. The weight is assigned based on the number of historical blockchains involved in each query. For a 10-year query, only 1 historical blockchain needs to be searched; therefore, the assigned weight is 1. Similarly, the weights are 2, 3, and 4 for 15-year, 20-year and 25-year queries, respectively.

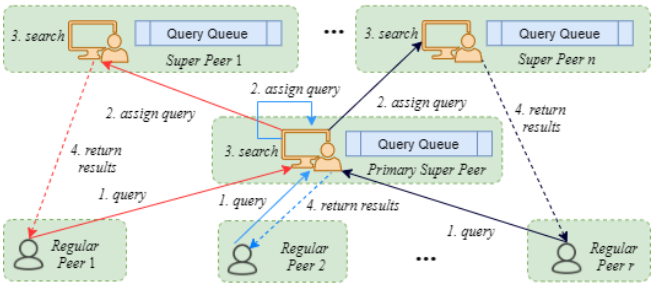

Figure 5. Querying process for accessing historical blockchain data

New queries are always first assigned to the super peer with the lowest total weight assigned. The queries sent to super peers are stored in their query queues, and the total weight of the queries assigned to each super peer must be

approximately equal. When a super peer's query queue is not empty, it performs a search to retrieve relevant historical data. A response is then returned to the querying regular peer, including a summary report of the relevant transactions, and links to associated files that the regular peer can download. Since the *PSP* is also a super peer, it assigns query requests to itself as well, and returns the query results directly to the requesting regular peers. Note that each super peer maintains its own copy of the historical blockchains; therefore, searches performed by all super peers can be completed concurrently.

B. Retrieval of Historical Blockchain Data

We now define the procedure how a super peer retrieves historical data for a query. Each query in the super peer's queue contains a patient ID and a search length in years as two input parameters. Let the current blockchain be one containing y years of data, where $5 \leq y < 10$. The first y years of each query can be done locally by the regular peer, since this data is stored in the current blockchain. When the search length is 10 years or more, at least one historical blockchain will need to be searched. This search will be facilitated by looking for indexes in the meta-blocks of the historical blockchains. Algorithm 2 shows how historical data can be retrieved from historical blockchains by a super peer.

Algorithm 2: Retrieving Historical Blockchain Data

Input: Patient ID *pID* and search length *sLen* in 5*x* years, $x \in [1, 5]$
Output: A summary report with all retrieved historical data

1. Create an empty summary report *SR*
2. Let *cDate* be the current date
3. **if** *sLen* == 5 **return** *SR* // search current blockchain only
4. **for each** historical blockchain *II*
5. Examine $MB_{II}.ED$ of meta-block MB_{II}
6. **if** $MB_{II}.ED < cDate - sLen$ // outside of the search period
7. **continue** // search the next historical blockchain
8. Get a list of triples *LTX* from $MB_{II}.HM$ with *pID* as the key
9. **for each** triple (*B*, *A*, *O*) in *LTX*
10. Read transaction *tx* from block *B* at address [*A*, *A* + *O*]
11. **if** $tx.TS \geq cDate - sLen$
12. Add retrieved *tx* and links to relevant files to *SR*
13. **return** summary report *SR*

As shown in the algorithm, by examining the *ED* in each meta-block, it will be known whether the associated historical blockchain should be included in the search. In each relevant meta-block, the patient ID in the query is used as the key to identify the relevant transactions and their exact locations in the historical blockchains. For each transaction, the super peer reads the transaction and adds the retrieved transaction to a summary report along with links to relevant files. Once completed, the summary report is returned to the requesting regular peer. Note that a regular peer can perform a local search for y years ($5 \leq y < 10$) in a similar manner, but only one meta-block of the current blockchain needs to be examined. For search lengths of 10 years or longer, the local search from the current blockchain and the remote search from the historical blockchain(s) are performed concurrently. Once the historical data is returned from a super peer, it is merged into the local search results by the regular peer.

V. Case Study

To illustrate the feasibility and the effectiveness of our approach, we conducted experiments and evaluated the performance of our scalable blockchain storage scheme based on the settings and results of each simulation. In our experiments, we assume that 10 large hospitals participate in a consortium blockchain network. One of the large hospitals is elected as a primary super peer, while the other 9 large hospitals serve as super peers. There are also 30 small and medium medical facilities in the network. We consider the lifetime of the blockchains to be at most 50 years because after 50 years, due to expected advances in computer technology, blockchain technology may be replaced by more advanced methods. We limit the total number of transactions in each block to 500, where each transaction may contain medical data in the form of image and text files. For simulation purposes, the number of visits per day is between [200, 500] and [50, 200] for large hospitals and small/medium-sized medical facilities, respectively.

A. Blockchain Size with an Annual Growth Rate

To determine the effects of the historical blockchain model, we used a time-based partitioning method to generate historical blockchains. In the model, super peers retain all historical blockchains as well as the current blockchain, while regular peers only need to store the current blockchain. Table 1 lists the parameters used in our experiments.

Table 1. Parameters used for blockchain size estimation

Probability of having images	Image size		Image count	
	Lower bound	Upper bound	Lower bound	Upper bound
5%	1 MB*	3 MB*	1	5

Probability of having text	Text Size		File size annual growth rate (%)	Time split
	Lower bound	Upper Bound		
100%	0.003 MB*	0.007 MB*	0, 1, 3, 5	10 yr**

* Initial values of bounds; all bounds are subject to increase by an annual file size growth rate.
** A time-based split occurs at 10 years; the earliest 5 years of data make an *HB*, while the latest 5 years of data are retained by the *CB*.

As shown in the table, we assume that for a hospital visit, the probability of having images, such as x-rays, attached to a doctor's notes is 5%. The size of the images is typically in the range of [1MB, 3MB] and the number of attached images is limited to 5. The sizes of text-based medical records are also listed in Table 1. Note that in our experiments, we consider annual file size growth rates of 0%, 1%, 3% and 5%. For example, when growth rate is 3%, the maximum image size can reach 13.15MB in 50 years, which is typically large enough for a medical image file.

We now simulate the creation of 50-year blockchains to estimate the sizes of blockchains. For each day, each large hospital or small/medium-sized medical facility in the

network generates a random number of visits within a given range. Each visit generates one transaction and is stored in a block that can store up to 500 transactions, regardless of transaction size. Each transaction has a 5% chance of including image file(s). If a transaction does include image file(s), the number of image files is chosen randomly within a given range. In addition, the size of each image file or text file is also randomly generated within a given range.

To deal with the possible year-to-year increase in image and text file size, we consider annual file size growth rate of 0%, 1%, 3% and 5% in our experiments. For each growth rate, data is collected from a sample of 10 simulations to establish average values for evaluation. A 0% growth rate is included as a baseline; while not a realistic assumption, this establishes the minimum size of the blockchain against which the other growth rates can be considered. Figure 6 and 7 show the changes in total blockchain size (including current and all historical blockchains) and the changes in current blockchain size along the years, respectively.

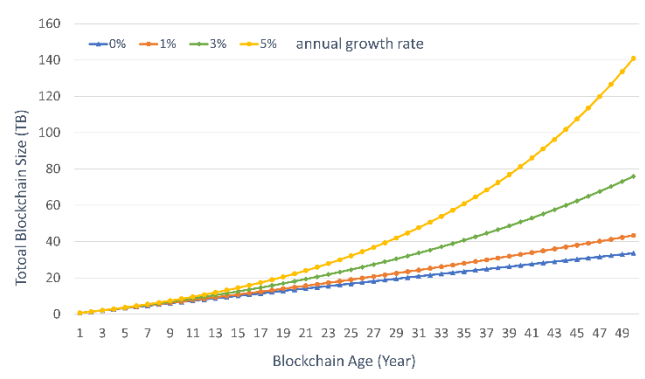

Figure 6. Total blockchain size by year with varying annual growth rates

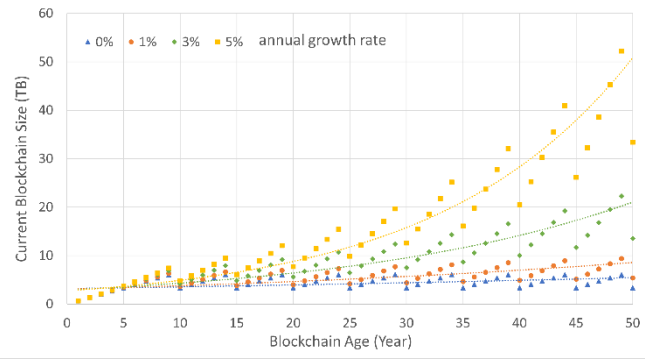

Figure 7. Current blockchain size by year with varying annual growth rates

Based on the experimental results, the effectiveness of using a historical blockchain structure is evident. After 50 years, the total blockchain size exceeds 33 TB at 0% growth, 43 TB at 1%, 76 TB at 3%, and 140 TB at 5%. For regular peers, storing the entirety of this data would become infeasible because the resources required would not make using a blockchain a practical storage solution for them. On the other hand, the storage load of regular peers can be greatly reduced if the historical blockchain structure is employed. The size of the current blockchain is far less than the size of the total blockchain data. At a 0% growth rate, the current blockchain size is at most 6 TB; at 1%, it is below 10 TB; at 3%, it is below 23 TB; and at 5% it is below 53 TB. Since these represent the maximum size of the current blockchain in 50 years, at all other times, the current blockchain is much smaller. We also note that the image file size is capped at nearly 34MB at 5% annual growth rate, which may be an overestimate. A 53 TB current blockchain is still larger than we would expect for the storage size of a regular peer, but we consider this outcome to be a worst-case scenario and unlikely to happen. Moreover, we can always limit the image size to ensure a reasonable size of the current blockchain that is maintained by regular peers. Meanwhile, with larger resource support, super peers can continue to store all blockchain data. Thus, by using the historical blockchain structure, we establish the use of blockchains as a viable storage solution for continuously growing data.

Note that we also considered a size-based partitioning strategy, where the size of the current blockchain is limited by a predefined parameter. For example, we allow splitting to occur when the current blockchain reaches 10 TB in size. In this case, the earliest 5 TB of data becomes a historical blockchain, while the most recent 5 TB of data remains in the current blockchain. However, our experimental results show that very few years of data can be stored in the current blockchain at a reasonable annual growth rate, which makes the current blockchain not useful enough for regular peers.

B. Data Retrieval Time for an Individual Request

In this experiment, we measure the data retrieval time for a regular peer to perform a query on the blockchain historical data. The data retrieval request is to search for a patient's medical records within a certain number of years. For any search within the current blockchain age, the data can be readily retrieved from the current blockchain; however, when the search time is greater than the current blockchain age, a query needs to be sent to a super peer to identify relevant data and retrieve it from the historical blockchain(s).

For each blockchain age, we simulate searches of a given length of years, which are 5, 10, 15, 20 and 25. If the search length is greater than the blockchain age, the search will stop at the end of the blockchain. For example, if 25 years of data is requested for a patient, but the blockchain is only 10 years old, only 10 years of data will be retrieved. In addition, for any search length of 10 years or longer, the first y years of data, where y is the current blockchain age, $5 \leq y < 10$, will be retrieved by the regular peer, and only the portion of the search greater than y years will be retrieved by a super peer. We choose the maximum number of years to be searched locally by the regular peer, because otherwise, a super peer must also search its current blockchain unnecessarily.

We use the same parameters listed in Table 1 for the image size bounds, image count bounds, text size bounds, and

probability of image occurrence in a medical record. However, we set the file size annual growth rate to 3%, which is more realistic than 5% as described in Section V.A. For search length of 10 years or more, measuring data retrieval time requires consideration of the *network latency time* for searching in the historical blockchain(s), *extraction time* for extracting index information from the relevant meta-blocks and the data from relevant blocks, and *data export time* for writing the extracted historical transaction data to a summary file. As outlined in Algorithm 2, the exact location of a transaction in a historical blockchain can be determined in constant time from the index information stored in the meta-blocks. However, it takes time to open a meta-block file and read data from it. Based on the average size of the meta-blocks, retrieving index information from a meta-block can take several seconds. Extracting transaction data involves reading the transactions from their locations, depending on the different sizes of randomly generated transactions. Finally, a super peer needs to create a summary file and send it to the requesting regular peer. Table 2 lists the additional parameters used for the data retrieval simulations.

Table 2. Parameters used for data retrieval simulations

Blockchain age (years)	Search length (years)	Patient visits (annual)		File size growth rate (annual)
		Lower bound	Upper bound	
10, 20, 30, 40, 50	5, 10, 15, 20, 25	1	7	3%

Network latency time	Extraction time	Data export time	Average meta-block size
0.5 seconds	0.02 s/MB	0.017 s/MB	200 MB

Note that access control policies will be utilized on the application level to ensure data privacy; a patient can only access their own data, while a healthcare provider can access any patient's data. Figure 9 shows the average of 1000 simulations for each search length and each blockchain age.

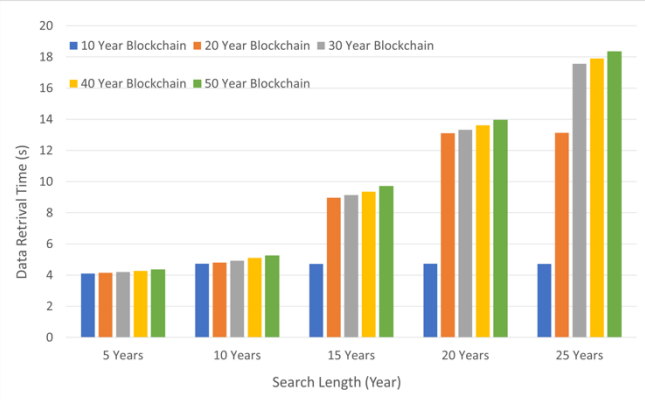

Figure 8. Data retrieval time for varying search length and blockchain age

From the figure, we can see that for the anticipated most common search lengths, i.e., 5 and 10 years, the search time averages less than 6 seconds for all blockchain ages. As the search length increases, the data retrieval time increases accordingly, with a maximum of about 18 seconds for a 25-year search length in a 50-year blockchain. Note that the 10-year search time does not increase significantly compared to the 5-year search time because the 10-year search consists of a local search by a regular peer in the current blockchain and a remote search of the remaining data by a super peer, both of which are performed concurrently. The insignificant increase in the average data retrieval time in the 10-year search is due to the search of historical data that results in the additional network latency time and data export time.

C. Data Retrieval Time for Concurrent Requests

Queries to historical blockchain data are handled by a group of super peers. In a group of 10 super peers, queries are assigned by the *PSP* to ensure even load balancing among the super peers. In this way, simultaneous historical blockchain data retrieval requests can be completed concurrently by the super peers. In our experiment, we expect most data retrieval requests in a network to be within 5 years, since the most relevant data in patient medical history is the most recent data. These retrievals can be completed by regular peers locally. To examine the search time of concurrent data retrieval requests, only requests from regular peers for 10 to 25 years of data are measured.

Concurrent search requests may occur within 5-minute intervals in a standard 8-hour workday. Since shorter searches are expected to be more common, we assign probabilities of 40%, 30%, 20% and 10% to the search lengths of a 10-year search, a 15-year search, a 20-year search and a 25-year search, respectively. We simulate 10, 20, 30, 40, and 50 concurrent searches at 5-minute intervals and calculate the total data retrieval times. While 50 concurrent requests represent a very high volume of requests in a 5-minute interval, this may occur at certain times of the year, such as flu season.

We calculate the average data retrieval time for a super peer to complete all the concurrent requests in its queue. For example, with 20 concurrent requests, each super peer is required to process about 2 concurrent requests in its request queue. Figure 10 shows the average data retrieval times for the specified numbers of concurrent requests and blockchain ages by running 1000 simulations.

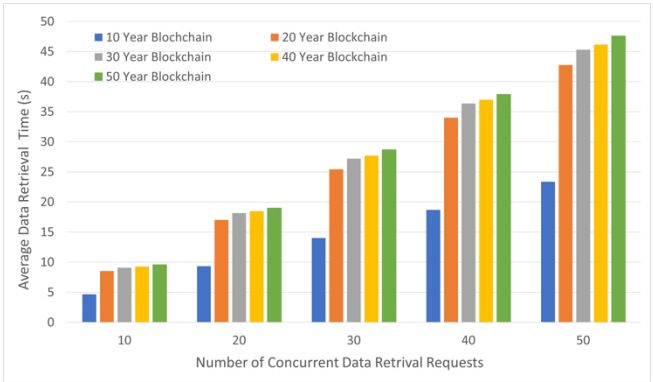

Figure 9. Average data retrieval time for concurrent data retrieval requests

From the figure, we can see that the average data retrieval time for 10-year blockchains is considerably lower than that for higher age blockchains. This is because a 10-year blockchain is unable to facilitate data retrieval beyond 10 years. On the other hand, the average data retrieval time for a 20-year blockchain is not considerably lower than those older blockchains, as 25-year searches only represent 10% of the total searches. For blockchains older than 20 years, increasing the number of concurrent searches by 10 results in approximately only 1 additional request per super peer in the query queue. Therefore, the corresponding increase in data retrieval time is equal to the average time to process one additional query request in a super peer's query queue. Overall, the average data retrieval time stays below 50 seconds, which shows that our concurrent search algorithm is feasible, even during periods of high accesses.

VI. Conclusions and Future Work

Scalability of blockchain has been a pervasive issue, and recent solutions have focused on moving data or computation off-chain by using IPFS and cloud-based storage structures. In this paper, we propose a novel approach to improve blockchain scalability while keeping all data on-chain. We introduce the concept of historical blockchain, where older sections of the current blockchain are separated after a specified time interval. This time-based partitioning strategy allows the current blockchain to contain a useful amount of relevant data, while freeing regular peers with short resource/storage from maintaining the entire data-intensive blockchain. The historical blockchains are maintained by a group of super peers with greater resources and computing power. In addition, we introduce a meta-block, attached to a historical or current blockchain, that serves as an index file and is used to facilitate efficient data retrieval. This block is not part of the blockchain and can be refreshed at any point in time by scanning the associated blockchain. Access to historical blockchain data is accomplished by regular peers through data retrieval queries sent to the primary super peer. These queries are then assigned to super peers using a predefined load balancing mechanism. The super peer collects the relevant data for the query by identifying the exact transaction locations found using the meta-blocks. Finally, a response containing a summary of the retrieved data is returned to the regular peer. Experimental results show that this approach can effectively reduce the storage burden of data-intensive blockchain applications on regular peers, while providing efficient access to historical data.

In future work, we will investigate how to improve the performance of concurrent data retrieval. One way is to analyze the effectiveness of parallel searches across multiple historical blockchains. This parallelization should allow a super peer to reduce search time if the historical blockchains are stored on different hard disks. We will also investigate effective methods for selecting the primary super peer based on feedback from regular peers. Alternatively, requests for historical data from regular peers can be broadcast to all super peers and a dynamic load-balancing algorithm can be developed to distribute concurrent query requests from regular peers evenly among the super peers. Finally, to ensure strong data privacy, it is necessary to design access control policies for users with different roles to access blockchain data with the required permissions [13]. This is particularly necessary in healthcare blockchain applications.

References

[1] S. Liu and H. Tang, "A Consortium Medical Blockchain Data Storage and Sharing Model Based on IPFS," In *Proceedings of the 4th International Conference on Computers in Management and Business (ICCMB 2021)*, January 2021, pp. 147-153.

[2] X. Chen, K. Zhang, X. Liang, W. Qiu, Z. Zhang, and D. Tuee, "HyperBSA: A High-Performance Consortium Blockchain Storage Architecture for Massive Data," *IEEE Access*, Vol. 8, September 2020, pp. 178402-178413.

[3] J. Poon and T. Dryja, "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments," *White Paper*, 2016. Retrived on September 1, 2022 from https://lightning.network/lightning-network-paper.pdf

[4] J. Eberhardt and S. Tai, "On or Off the Blockchain? Insights on Off-Chaining Computation and Data," In: De Paoli, F., Schulte, S., Broch Johnsen, E. (eds) *Service-Oriented and Cloud Computing*, ESOCC 2017, Lecture Notes in Computer Science, Vol. 10465. Springer, Cham, pp. 3-15.

[5] G. Wang, Z. Shi, M. Nixon, and S. Han, "ChainSplitter: Towards Blockchain-Based Industrial IoT Architecture for Supporting Hierarchical Storage," In *Proceedings of the 2019 IEEE International Conference on Blockchain (Blockchain)*, July 14-17, 2019, Atlanta, GA, USA, pp. 166-175.

[6] D. Li, W. E. Wong, M. Zhao, and Q. Hou, "Secure Storage and Access for Task-Scheduling Schemes on Consortium Blockchain and Interplanetary File System," *IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, IEEE BSC 2020, Macau, China, December 11-14, 2020, pp. 153-159.

[7] Y. Xu, "Section-Blockchain: A Storage Reduced Blockchain Protocol, the Foundation of an Autotrophic Decentralized Storage Architecture," In *Proceddings of the 23rd International Conference on Engineering of Complex Computer Systems (ICECCS)*, 2018, pp. 115-125.

[8] Y. Xu and Y. Huang, "Segment Blockchain: A Size Reduced Storage Mechanism for Blockchain," *IEEE Access*, Vol. 8, January 2020, pp. 17434-17441.

[9] A. Thamrin and H. Xu, "Cloud-Based Blockchains for Secure and Reliable Big Data Storage Service in Healthcare Systems," In *Proceedings of the 15th IEEE International Conference on Service-Oriented System Engineering (IEEE SOSE 2021)*, Oxford Brookes University, UK, August 23-26, 2021, pp. 81-89.

[10] J. Jayabalan and N.Jeyanthi, "Scalable Blockchain Model Using Off-chain IPFS Storage for Healthcare Data Security and Privacy," *Journal of Parallel and Distributed Computing*, Vol. 164, 2022, pp. 152-167.

[11] H. Im, K. H. Kim, and J. H. Kim, "Privacy and Ledger Size Analysis for Healthcare Blockchain," In *Proceedings of the 2020 International Conference on Information Networking (ICOIN)*, 2020, pp. 825-829.

[12] A. Thamrin, H. Xu, and R. Ming, "Cloud-Based Hierarchical Consortium Blockchain Networks for Timely Publication and Efficient Retrieval of Electronic Health Records," *Advances in Science, Technology and Engineering Systems Journal (ASTESJ)*, Special Issue on Multidisciplinary Sciences and Engineering, Vol. 7, No. 2, April 2022, pp. 179-190.

[13] H. Guo, W. Li, M. Nejad, and C. Shen, "Access Control for Electronic Health Records with Hybrid Blockchain-Edge Architecture," In *Proceedings of the 2019 IEEE International Conference on Blockchain (Blockchain)*, July 14-17, 2019, Atlanta, GA, USA, pp. 44-51.