SWARMED: A HIGH-THROUGHPUT INTEROPERABILITY ARCHITECTURE OVER ETHEREUM AND SWARM FOR BIG BIOMEDICAL DATA

Arghya Kusum Das

Department of Computer Science, University of Alaska Fairbanks, Fairbanks, USA

ABSTRACT

In this paper, we introduce SwarMED, a decentralized yet high throughput interoperability system for big biomedical data. SwarMED uses Etehreum blockchain for trustless security and Swarm p2p storage to handle high throughput transaction of big data. In SwarMED, we developed an indexing mechanism over the immutable storage of Swarm to achieve high-throughput while sharing millions of patient records and images among multiple parties.

SwarMED achieved a high throughput of 250K medical records per second over a private network constructed over LSU-HPC cluster. This high throughput is 9x more comparing to conventional way of using p2p storage in conjunction with blockchain. This high throughput enables the patients to get real-time access to his comprehensive medical history and scientists to gain real-time access to different medical data for collaborative research complying to the constraints posed by existing laws.

Our system-level analysis over different design alternatives over different transfer and storage architectures shows that, p2p storage platforms automatically provide significantly better scalability over traditional HTTP with increasing number of clients. Swarm provides 2x more I/O throughput and 10x less latency than IPFS, another p2p storage system making it a better choice for decentralized big data transaction

KEYWORDS

Blockchain, Decentralized storage, Ethereum, Swarm, Big data, Biomedical

1. INTRODUCTION

Blockchain and its distributed ledger technology received a significant attention in the health care sector recently. Starting from the issues in data-interoperability to counterfeit medicine, Blockchain showed its promises in terms of a tamper-proof, decentralized, trust-less, immutable, secured solution. Consequently, the secured, distributed ledger of different Blockchain network (e.g., Ethereum, Hyperledger, etc) is increasingly used in different field of healthcare.

For example, [1] envisions the use of immutable audit trail of Blockchain to fight counterfeit medicine. [2] shows how the decentralized trust model of Blockchain can be used to improve the relationship between the patients and the physicians. [3], etc. envision the Blockchain's use to improve the existing EMR-management-system. Moving a step forward, in response to the ONC's nation wide challenge [4], many pragmatic technical architectures have been proposed to improve the existing infrastructure of EMR-management and data-interoperability. For example,

[5] uses Ethereum to provide the patients an immutable log of their medical history. [6] developed decentralized building blocks for data sharing and interoperability among multiple party using Hyperledger Blockchain. [7, 8], etc. also proposed their own interoperability architecture to share medical records using their own proprietary Blockchain networks.

Although common and crucial for medical records, data management capability is severely lacking in these frameworks. The Proof of Work (PoW) consensus and the random disk access technique of vanilla Blockchain in each individual machine of the network significantly limit the throughput of these above-mentioned frameworks especially when the chain's size grows to accommodate terabytes of big data.

In this paper, we propose SwarMed which uses Ethereum Blockchain for data-interoperability. However, to address the performance limitations of vanilla Ethereum, we use Swarm, a peer-to-peer storage system for data management. Unlike Etherem, in our system Swarm does not use PoW and it reads/writes big data to the individual's disk sequentially which improves the throughput of Ethereum. Our architecture shows uniform performance with increase in the number of users. Comparing to the existing HTTP-based architecture, we observed more than 4x gain in performance.

We also developed an indexing mechanism to further improve the throughput of our architecture. We observed another 9x performance gain comparing to the naive way of storing all the address pointers for the data on chain.

The rest of the paper is organized as follows:

Section 2 describes the related technologies such as Ethereum and Swarm. In Section 3, we discuss the data-interoperability architecture. Section 4 evaluates our architecture with several other design alternatives. In Section 5 we evaluate SwarMed with the ONC's interoperability road map. Finally, Section 6 concludes the paper.

2. BACKGROUND

2.1. Ethereum Blockchain and Smart-Contract

The core of Ethereum platform is an ecosystem for developing dApps (Decentralized Applications) whose main purpose is to manage and execute Smart contracts which builds the business or project logic on the top of the Blockchain network. The back-end of a dApp contains EVM (Ethereum Virtual Machine), which run bytecodes of contracts and is responsible for communicating with other nodes in the network using peer-to-peer networking.

2.2. Swarm

Swarm provides a peer-to-peer storage system where data is stored based on a hash calculated from its actual content. The data is replicated over many servers. However, they can be retrieved by their content-hash only without knowing the server information. Consequently, the data can be downloaded from the nearest peer of the P2P system, rather than a specific server address located significantly many more hops apart in a traditional system. It results in a significantly higher throughput and scalability during the data download. In addition to that, these P2P storage systems are automatically scalable as each of the peer or the clients works as a storage server also and they are committed by means of some incentive mechanism of the system.

3. SWARMED ARCHITECTURE

Figure 1 shows the architectural overview of SwarMED. In this proof-of-concept version, we developed SwarMED as an interoperability layer isolated from the individual's database or file system. That means, the user has the full control over how much and what types of data are to be shared.

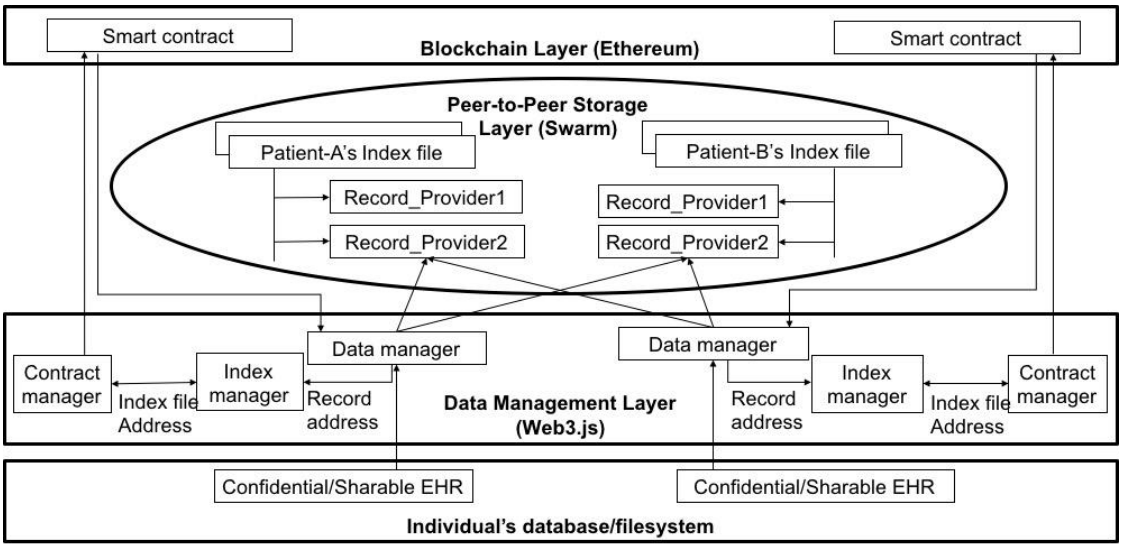
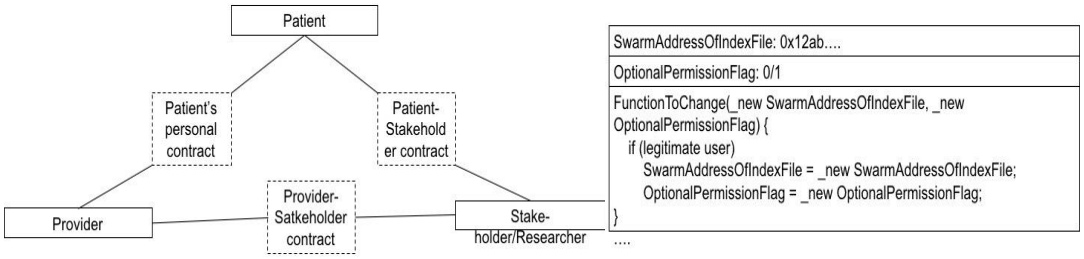


Figure 1. Decentralized Architecture

SwarMED architecture has three different layers such as, blockchain layer which takes care of the consensus protocol, peer-to-peer storage layer and data management layer. Following are the description of each of the layers. For the sake of brevity, we mainly focus on the performance issues at system level and avoid the intriguing details of the smart contracts that take care of the legal consents.

3.1. Blockchain layer



a. Different types of smart contracts b. Minimal structure of a smart contract

Figure 2. Smart Contract

This layer implements a set of smart contracts providing the full functionality required to join and participate in the blockchain network. For better understandability, we first divided the entire medical domain into three different entities such as, patient, provider and other third party stake holders. Then we present the Ethereum's smart contracts as the data-sharing relation between the three entities of SwarMED as shown in Figure 2a.

Although smart contracts are responsible for all logistic issues, we show only its data handling part in Figure 2b. As shown in this minimal structure, every smart contract stores only one Swarm hash of 32bytes to points to the actual medical records. This hash is updated as more data is added for the patient.

3.2. Peer-to-Peer storage layer

The major objective of using peer-to-peer storage is to serve a solution for big data sharing that is DDOS-resistant, zero-downtime and fault-tolerant. Medical records are stored in separate node in the form of small chunks and stay distributed. The Blockchain network of Ethereum contains only the address (Swarm hash) of the data keeping the Blockchain lightweight thereby, improving the throughput.

3.3. Data Management Layer

This layer consists of a three different modules: 1) Index Manager, 2) Data Manager, and 2) Contract Manager. However, for the page limitation, we focus mainly on the Index Manager as it is the main module that improve Ethereum's throughput when handling big data.

3.3.1. Index Manager

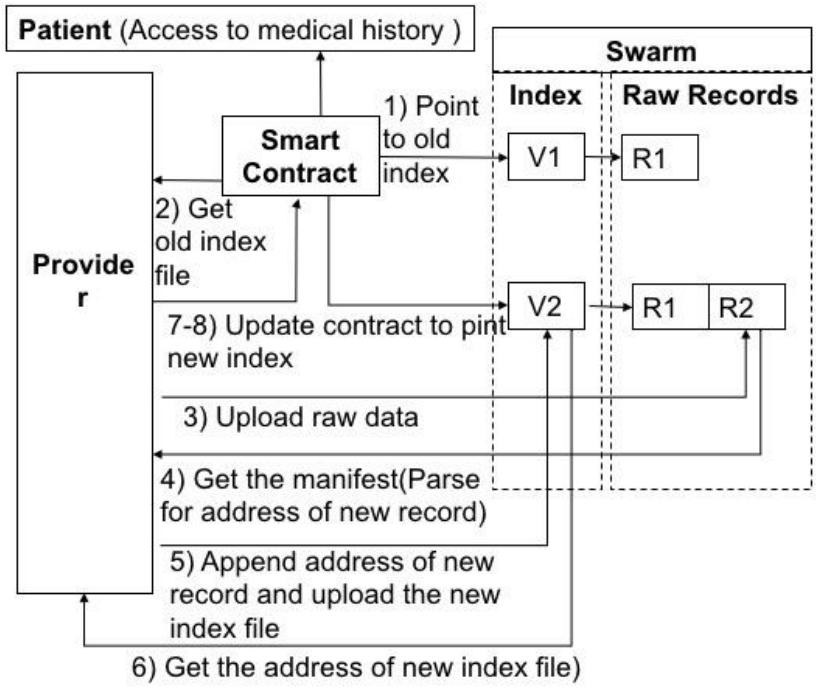


Figure 3. Indexing Service

Index manager implements a query interface for the the raw medical data that are uploaded in Swarm. We developed the index manager such a way so that the contract keeps the metadata of metadata to the raw records which is only a single address (called swarm-hash) that points to an index file kept in Swarm.

Broadly, the index manager collects the Swarm-hashes of raw medical records in Swarm through the upload-manager and update those in an index file in timely fashion. As mentioned earlier,

Swarm is an immutable peer-to-peer storage, i.e., the file written once in Swarm cannot be updated in place. Hence, in this context update means creating a new version of the file with updated content (as shown in Fig. 3). Once the new version of index file is created in Swarm, its address is updated in the contract. Now, that the patient obtained the indexes of his medical records from the contract he can access it using bzz protocol (or its variation, such as bzzi and bzzr) from the Swarm-gateway. The file cannot be accessed without this swarm-hash.

Figure 3 shows the update procedure for the immutable index file in Swarm and the contract in the blockchain. Initially, the contract holds the pointer of an older version of the index file kept in Swarm. For a new patient registered in the system the contract points to an empty index file. The provider gets an old list of swarm-hashes (or, an empty list for a new patient) by accessing the contract between the patient and the provider. Gradually, the providers start adding patient's medical records to swarm and populate the index file with the corresponding swarm-hashes. Each time the a provider upload the medical record to Swarm, Swarm returns the manifest of these data files (or, directory). We parse the manifest to get the swarm-hash value (i.e., the location) of the newly uploaded files (or directory). The provider appends this swarm-hash with the old list of swarm-hashes obtained from the old version of the contract and create a new index file. Finally, the new index file is uploaded to Swarm and notify the contract manager with the swarm-hash of this new index file.

It is to be noted, that the providers upload the raw medical record for their patient only once in Swarm (similar to the existing database system). Based upon the data-sharing-agreement defined in the contract an index file is created pointing to the required subset of records and the contract is updated with the address (swarm-hash) of the index file only. This way, SwarMED keeps the block-size small and constant on the main blockchain also avoids any data duplication in the Swarm peer-to-peer storage.

3.3.2. Data Manager

This module provides the only access interface to the node's local database or file system. Any type of data definition routines including encryption (e.g., SHA, AES, etc), access model (e.g., PCORNet, OMOP, etc.) can be implemented here.

3.3.3. Contract Manager

It provides the interface between the index manager and the Ethereum contracts. It keeps track of all the patient's contract in the system and access the corresponding contract when required. Any read/write between the Index Manager and the Etehreum smart contract takes place through this module.

4. EVALUATION

We evaluate SwarMED at its component level as well as overall design level considering many different alternatives.

4.1. Data

For this paper, we use a synthetic and anonymous data set of 80million patient records following the PCORNet Common Data Model (CDM) which allows for the systematic analysis of disparate observational databases. We have identified 140 different attributes for a patient and simulated

the records. Each records is 1.5KB in size. Like the real world, each patient is assumed to visit many providers many times in his life span.

4.2. Compute Environment

For the evaluation purpose we constructed a private Ethereum Blockchain over the LSU HPC cluster called SuperMic. As shown in Table \ref{tab_cluster}, each node of SuperMic has 2 Intel IveBridge Xeon processor with 10 cores each yielding a total of 20cores per node. Each node has 64GB of DRAM and one hard disk drive (HDD) attached. The throughput of the disk is evaluated to 160MB/s which means a total of 106667 records from our data set can be written to the disk per second.

Table 1. Compute Cluster.

Text	**Alingnment**
Total number of nodes	16
Processors/node	2 Intel ivebridge
Number of cores/node	20
Storage/node	1 HDD
Disk throughput/node	160MB/s
DRAM/node	64GB

4.3. Design Alternatives Evaluated

The following five design alternatives have been evaluated to show the relative merits of the architecture:

4.3.1. Traditional HTTP- Storage

In this case, the patient's medical data is stored in a standard HTTP server. It is the most commonly used infrastructure to transfer data over the Internet including the cloud-based architectures also.

4.3.2. P2P Storage (Swarm and IPFS)

The entire patient dataset is stored on the P2P storage and accessed via its hash-based guarantee of data integrity. The clients can join and leave the network any time they wish. Unlike HTTP, the data is replicated over multiple clients automatically when they join the network and is downloaded from the nearest source possible. Swarm and IPFS are evaluated individually to select the most sustainable architecture for the big data transfer.

4.3.3. Blockchain Storage

This scenario uses the smart contract storage to store the patient records providing immutability and reliable time stamping on the data itself. Avoiding the need to manage a separate data store, this solution stores the data in smart contract permitting the checking of individual patient record.

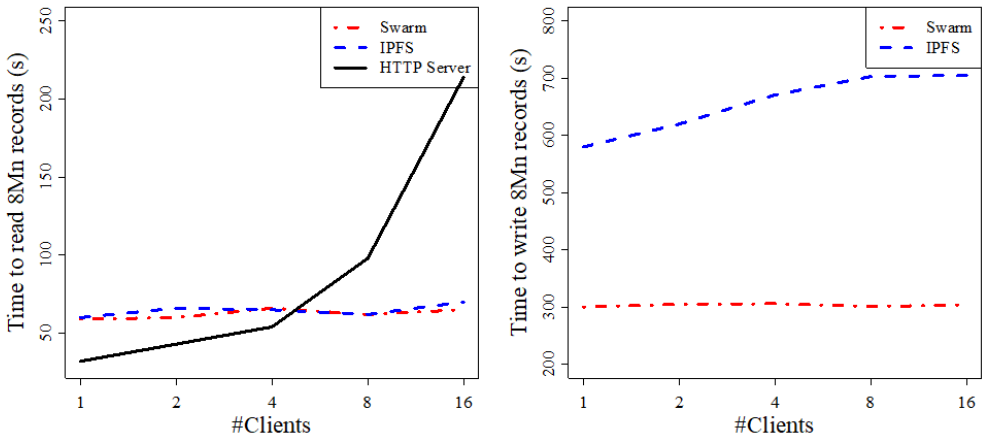
4.3.4. Blockchain + HTTP- Storage

In this design, the query string and the server address are stored in the smart-contract of Ethereum. On successful execution of the smart-contract, the data is fetched from the HTTP server in a traditional HTTP-based manner.

4.3.5. Blockchain + P2P- Storage

This is similar to the previous one but instead of the HTTP-based query string or server address the content-hash of the data is stored in the Ethereum smart-contract. In this design, all the hashes are stored in the Blockchain providing the immutability guarantee at the dataset level. The SwarMed architecture discussed earlier in \ref{sec_SwarMEDArchitecture} basically an enhancement over this design alternative.

4.4. Transferring Big Data over P2P and HTTP



a.Reading 8Mn patient records b. Writing 8Mn patient records

Figure 4. Comparing Swarm, IPFS and HTTP

Figure 4a compares the performance of P2P- and HTTP-based storage. Because of multiple replication strategy and hash-based read (download) strategy from the nearest node, P2P clients (or servers) show a uniform performance when the number of clients increases. On the other hand, HTTP's performance degrades severely with the growing number of clients clearly showing the scalability issues. Hence, we inferred that the P2P storage is provides better solution for health care interoperability where millions of users reads (download) data over Internet every single day.

4.5. Swarm vs IPFS

Figure 4b compares the performance Swarm and IPFS in terms of data write (or, upload). We evaluate both the storage system in terms of strong scalability. That is one Swarm or IPFS node writes 8million unique patient records to the cluster with varying cluster size. Swarm shows more than 2x performance gain comparing to IPFS. Furthermore, we observed a slight increase in the execution time of IPFS's write with increase in number of nodes whereas Swarm shows similar performance throughout.

Although Swarm and IPFS both shows similar performance in terms of read (Figure 4a), Swarm outperforms IPFS in terms write. As the healthcare interoperability is both read- and write-heavy in nature we used Swarm in stead of IPFS as a more sustainable solution

4.6. Blockchain Performance

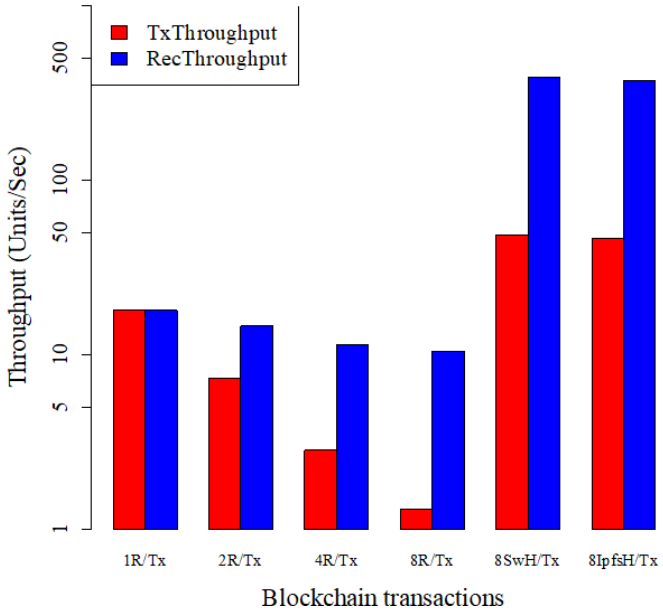


Figure 5. Blockchain performance

Figure 5 shows a linear loss (First 4 groups of the Figure 5) in blockchain's performance with increasing size of data. Although the average size of each record is only 1.5KB in our experiments, blockchain's transactional throughput shows sharp decline with increasing number of patients' records per transaction even-though there is less than 10 records (i.e. 15KB only) per transaction. In terms of record-throughput, i.e., the number patient's records can be written and transferred through the main chain per second also decreases.

4.7. Blockchain + HTTP

Although this design approach is adopted in many architecture (e.g., [9]), it should be remembered that many centralized server is not same as a decentralized server-based architecture. Any of these centralized server can be attacked individually hampering the service from that subset of data. Consequently, this design does not provide any security against DDoS or fault tolerance which are the main purpose of a Blockchain-based system. Consequently, we did not select this design and eliminate from our performance evaluation.

4.8. Blockchain + Swarm

As discussed earlier, in this design consideration, the data resides inside a P2P storage and the content-hash (64Bytes) is stored in the blockchain transaction. The system is tamper-proof, free from DDoS attack and at the same time offload the data from the main chain to guarantee the throughput.

The last two bars of Figure 5 compare the performance of this design to the blockchain-only design. As the data size on the transaction decreases the gas used per transaction also decreases in this current design. Consequently, many transactions are accumulated in a block and mined simultaneously. Hence, in a busy time when many transactions enter the blockchain system from many client, the throughput will be significantly more. As shown in Figure 5 this design

alternative produce 25x better throughput in terms of number of blockchain transaction per second comparing to blockchain alone. In terms of number of records transferred per second the corresponding gain is 21x.

Swarm always outperforms IPFS both in terms of read and write. However, in conjunction with Ethereum, both perform similarly because of the performance is bottlenecked by Ethereum Blockchain. Although we use Swarm for its visible and immediate advantage, both Swarm and IPFS are changing rapidly. Hence, we use the P2P storage as a loosely coupled layer so that it can be changed with minimum effort.

4.8.1. Scalability of the Design

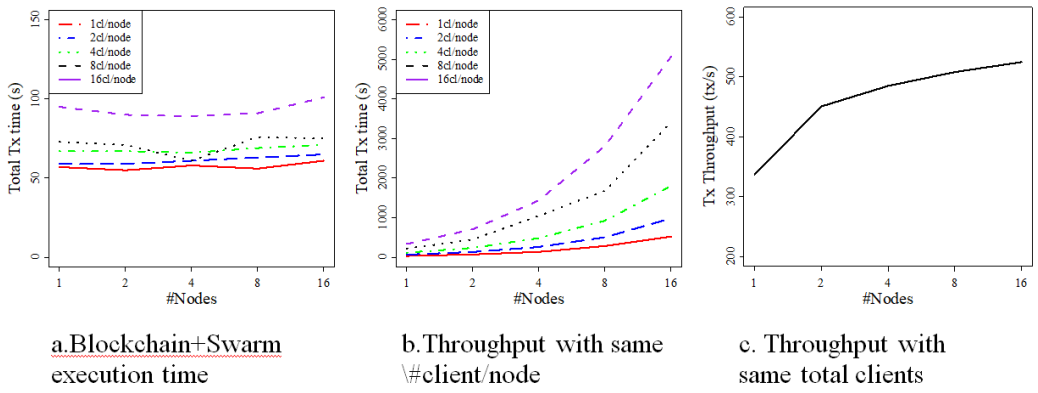
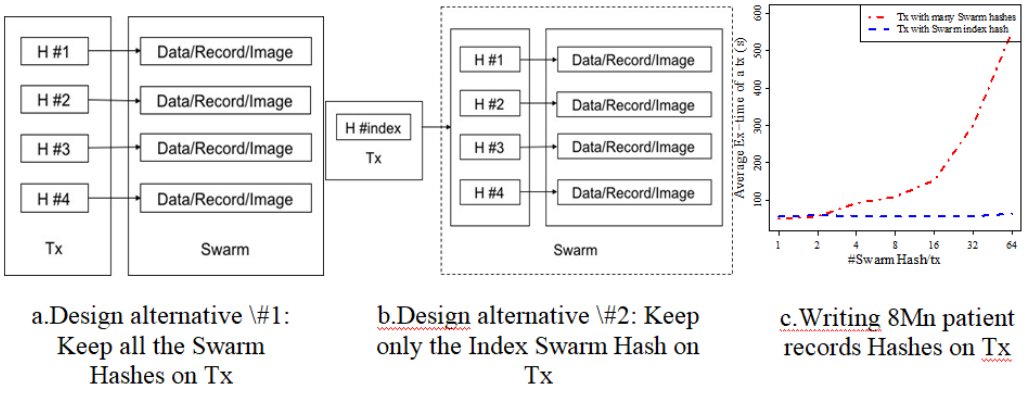


Figure 6. Blockchain+Swarm Write scalability

Figure 6 shows the scalability of the design. We store one swarm hash per blockchain transaction pointing to a single patient record kept in its P2P storage. To simulate the busy scenario, we have assigned multiple clients for each Ethereum node where each of the client sends multiple transaction request at once. That is, at a single time slot (less than a second) the total number of transactions in the system can be given by $\#clients * \#requests * \#nodes$. In these set of experiments, we keep $\#requests = 2000$. When there is 16 nodes, 16 clients are assigned to each node there are $16 * 2000 * 16 = 512000$ transaction requests are in the system.

As it can be seen in Figure and 6a, in a busy scenario with multiple clients and thousands of requests the design is weekly scalable for both read and write operations (i.e., upload and download operations). That is, the execution time remains almost similar with increasing number of Ethereum nodes when the number of clients (alternatively number of transaction requests) per node also increase at the same proportion. Each Swarm hash points to only one patient record in this set of experiments yielding the same record throughput per transaction.

A direct interpretation of this result shows the throughput of the system when keeping the number of the client same per node. As it can be seen in Figure 6b and 6c, the architecture shows scalable behaviour for both same number of clients per node as well as same number of clients in the entire system.



a.Design alternative \#1: Keep all the Swarm Hashes on Tx

b.Design alternative \#2: Keep only the Index Swarm Hash on Tx

c.Writing 8Mn patient records Hashes on Tx

Figure 7. Comparing Blockchain+Swarm design alternatives

4.9. Blockchain + Swarm + Indexing

Although Swarm in conjunction with blockchain shows significantly better performance comparing to the blockchain alone, the performance of the system can be bottlenecked with number of hashes stored in transaction. Figure 7a and 7b shows two different design alternatives with different number of hashes stored on the chain although achieve similar functional result.

To evaluate the benefit of the indexing quantitatively, we first created a 25MB files including 16700 records per file. Although this many records per patient is not common in the real world, the byte size reflects the presence of x-ray images, mammogram images, etc. For each of these files a Swarm hash is written on the transaction. Since the major bottleneck is observed in the data size in Blockchain transaction and not in the Swarm, the experiment reflects the real-world scenario giving a good quantitative metric to express the capability of the system.

To pinpoint the benefit of our design, we assign 1 client per node of a 16 node Ethereum cluster each working as a Swarm peer also. Each client sends a 200 trasaction request. The total amount of data migrated through the system can be given by $\#nodes * \frac{\#clients}{node} * \frac{\#requests}{client} * swarmFileSize$. That is, for a 16node cluster, 1 client per node with 200 requests per clients, a total of 80GB ($16 * 1 * 200 * 25MB$) data is migrated.

Figure 7c compares both the design. As it can be seen, the average execution time of the first design alternative i.e., many swarm hashes on the transaction increases exponentially with increase in number of swarm hashes. On the other hand, the index-based design performs almost similar for any number of Swarm hashes as the blockchain is kept light weight always.

The record-throughput of both the system can be calculated as $(\frac{\#records}{client} * \frac{\#clients}{node} * \#nodes)/averageExecutionTime$. For a 16 node cluster, 1 client per node, 16700 records written by each client, the fist alternative shows a throughput of 31010.60 ($16700 * 1 * 16 * 64 / 551.45$) records/s where as the proposed index-based design shows 9x performance gain yielding a throughput of 267033.10 ($16700 * 1 * 16 * 64 / 64.04$) records/s.

5. EVALUATING SWARMED WITH ONC'S INTEROPERABILITY ROADMAP

The major focus of SwarMed is to share the data among multiple stake holders in the health care domain. By sharing the data from different providers to a patient, SwarMed directly addresses the ONC interoperability roadmap's principal clause: "Individuals have access to longitudinal

electronic health information, can contribute to the information, and can direct it to any electronic location" [5].

SwarMed uses decentralized, trust-less technologies for data interoperability. P2P storage is tamper-resistant by means of its content hash, whereas, Blockchain provides a layer of encryption along with a time stamp information for any addition or modification of the data. Furthermore, the proposed architecture is fault tolerant and free of DDoS attack. Consequently, SwarMED is able to address the ONC's requirement for "secure and trusted exchange of electronic health information, consistent with privacy protections and individuals' preferences, across states, networks, and entities".

6. CONCLUSION

We proposed SwarMED, a scalable, high throughput architecture to share large-scale medical data over the Ethereum blockchain. We developed an efficient way of accessing data off-the-chain using Swarm in conjunction with our own indexing mechanism and surpasses the current throughput limitation of blockchain by several orders of magnitude. We also develop an indexing mechanism over Swarm, which provides an initial starting point for a P2P storage database that can improve the throughput of blockchain-based system for handling big data.

REFERENCES

[1] M. Mettler, "Blockchain technology in healthcare: The revolution starts here," in e-Health Networking, Applications and Services (Healthcom), 2016 IEEE 18th International Conference on. IEEE, 2016, pp. 1–3.

[2] A. Gropper. (2016) Powering the physician-patient relationship with hie of one blockchain health it. [Online]. Available: https://www.healthit.gov/sites/default/files/7-29- poweringthephysician-patientrelationshipwithblockchainhealthit.pdf.

[3] T.-T. Kuo, H.-E. Kim, and L. Ohno-Machado, "Blockchain distributed ledger technologies for biomedical and health care applications," Journal of the American Medical Informatics Association, vol. 24, no. 6, pp. 1211–1220, 2017.

[4] D. J. Bodas-Sagi and J. M. Labeaga, "Big data and health economics: Opportunities, challenges and risks," International Journal of Interactive Multimedia and Artificial Intelligence, no. In Press.

[5] T. O. of the National Coordinator for Health Information Technology (ONC). (2015) Report on health information blocking. [Online]. Available: https://www.healthit.gov/sites/default/files/reports/info blocking 040915.pdf.

[6] A. Ekblaw, A. Azaria, J. D. Halamka, and A. Lippman, "A case study for blockchain in healthcare:medrec prototype for electronic health records and medical research data," 2016.

[7] I. G. B. S. P. S. Team. (2016) Blockchain: The chain of trust and its potential to transform healthcare our point of view. [Online]. Available: https://www.healthit.gov/sites/default/files/8- 31-blockchain-ibm ideation-challenge aug8.pdf.

[8] R. Krawiec, D. Housman, F. M. White, Mark, F. Quarre, D. Barr, A. Nesbitt, K. Fedosova, J. Killmeyer, A. Israel, and L. Tsai. (2016) Blockchaina new model for health information exchanges. [Online]. Available: https://www2.deloitte.com/content/dam/Deloitte/us/Documents/publicsector/us-blockchain-opportunities-for-health-care.pdf.

[9] C. Brodersen, B. Kalis, C. Leong, E. Mitchell, Eand Pupo, and A. Truscott. (2016) Blockchain: Securing a new health interoperability experience. [Online]. Available: https://www.healthit.gov/sites/default/files/2- 49-accenture onc blockchain challenge response august8 final.pdf.

AUTHOR

Arghya Kusum Das is an Assistant Professor of Computer Science at the University of Alaska -Fairbanks. He earned his Ph.D. in Computer Science with an emphasis in Scientific Big data Analysis from Louisiana State University, USA. Dr. Das's research focuses on the scalable and efficient analysis of scientific big data (e.g., genomics, metagenomics and healthcare data). It includes both development of large-scale software frameworks and proposing new hardware architecture for efficient data analysis. His research also focuses on the secured transfer of big data over the Internet using Blockchain and other decentralized technologies. Das is currently working towards developing an innovative, online platform, and curriculums to spread quality education in the field of big data analysis, AI, HPC and related technologies.