Data Analysis in Blockchain

(Master's Dissertation Extended Summary)

Miguel Baptista, Miguel Mira da Silva, Paulo Rupino da Cunha

Abstract—Blockchain and Data Analysis are two topics of high interest, and both are being studied and integrated for multiple applications. Blockchain's main benefit is creating trust, security and privacy in a digital environment to gather data from different sources. Blockchain's main challenge is the time necessary to access and analyze stored data, and lack of tools to do so. In this work, a systematic literature review and a novel architecture are proposed. The main conclusions of the review were, using distributed file systems can avoid high storage and computation costs; the most common way of accessing data in blockchain, although sub-optimal, is smart contracts. With the gathered knowledge, a novel architecture was developed. Besides blockchain's and distributed file systems joint inherent capabilities, the architecture main benefit is the ability to make fast and up-to-date predictions using incremental machine learning. A proof-of-concept demonstrating its use was also implemented using Hyperledger Fabric (Blockchain), the InterPlanetary File System (DFS), Kafka (Distributed Data Streaming Event Platform) and RiverML (Incremental Machine Learning). The system was evaluated showcasing its scalability and cross-applications potential. Laslty, main contributions, related work, research limitations and future work are presented.

Index Terms—Blockchain; Information System Security; Data Analysis; Data Streams; Incremental Machine Learning; Distributed File System.

1 Introduction

Recently there has been an increase in the number of studies about blockchain-based technology and its applications in multiple fields, due to its ability to create trust in a digital environment [1].

Blockchain is a distributed tamper-resistant append-only ledger. Data is organized in blocks that are "linked" to previous ones via hashes. These hash pointers are created using the previous block as input on a hash function. When adding a new block, its validity is verified among the blockchain participants, and through the consensus algorithm, they agree (or not) on adding this new block to the blockchain.

Blockchain's architecture provides a system where the change of a previously added component is not allowed, making it immutable since any change is identified as a malicious attack and is not accepted by the network. This way, blockchain technology creates trust between its participants due to its secure and irreversible storage. More recent blockchains support smart contracts, "programs that implement the automated processing of traditional contracts" [2]. These programs execute automatically whenever previously agreed conditions are met.

The immutability properties of blockchain create a high volume of data to store making the cost of maintaining the network and appending new blocks expensive over time or when scaling up the network. Distributed File Systems were introduced as a solution to tackle this problem. DFS are peer-to-peer data networks that can be described as a network of systems capable of data storage, replication, distribution, and exchange [3]. By combining DFS and blockchain technology, DFS data integrity issues are solved through blockchain, as blockchain high maintenance costs regarding storage are addressed.

Data analysis can generally be described as the process of information discovery from data. For this to be possible, data needs to be collected, accessed, processed and finally analyzed. By finding patterns on the data during the analysis phase, data can be transformed into information. Recently, data analysis has also increased in popularity due to machine learning. Machine learning is capable of building models based on large amounts of data. The created models improve data's utility, which has been of great value for a different number of industries.

Accessing data on a blockchain is not a process as straightforward as on a regular centralized data repository, like a database. Blockchain does not have a built-in query system and most solutions are divided in one of two categories: emulating querying, with smart contracts and custom search engines, or extracting the data to a traditional database and access it from there. However, both solutions have problems. Querying data through smart contracts has high cost and slow performance. Extracting data to an off chain database violates the data integrity, since the data is no longer on chain. Adding DFS to the process only increases the complexity of the problem.

The objective of this research is to improve the time it takes to perform data analysis in blockchain based systems. Through the use of an architecture that also uses distributed file systems it is possible to reduce storage costs. As such, the second goal of this work is to create an architecture that also improves blockchain's storage costs with a DFS. The most common tool used for analyzing large amounts of data is machine learning so it should also be integrated in the mentioned architecture. Lastly, a proof of concept should be created to demonstrate the architecture usage.

In Section 2 the research methodologies chosen to conduct this work are presented. Section 3 showcase the first research methodology usage and answers three research questions; (3.3.1) Which distributed file systems are used with blockchain? (3.3.2) How is data accessed on architectures using blockchain and distributed file systems? (3.3.3) Which are the current streaming data architectures used in blockchain? Section 4 presents the research results application with a novel architecture. To demonstrate its usage, a software proof-of-concept is developed and evaluated. Lastly, Section 5 makes the conclusion remarks and presents the main contributions, research limitations and future work of this study.

2 Research Methodology

In this section, the research methodologies chosen to conduct this paper are presented. The research outline is also presented.

2.1 Systematic Literature Review

A Systematic Literature Review (SLR) is defined as a "means of identifying, evaluating and interpreting all available research relevant to a particular research question, or topic area or phenomenon of interest" [4].

In order to answer the research questions a systematic literature review was chosen since it is a trustworthy research methodology and it is useful to summarize and organize the investigation done in the field of blockchain data analysis. By performing a SLR we are able to identify any gaps in the topic while establishing the framework for the investigation.

The SLR conducted was based on Kitchenham 2004 study [4] and comprises three steps: planning, conducting and reporting. The planning phase is composed of the following three tasks; identify why the review is needed, develop a review protocol and define the research questions. The conducting phase is divided in two parts; screen and select the target studies and analyze the studies data. Lastly, the reporting phase purpose is to summarize the information gathered in the studies. The SLR aim is to identify the problem and get answers to our proposed research questions.

2.2 Design Science Research

The Design Science Research Methodology (DSRM) is the chosen methodology to guide this research since it provides rigorous guidelines for the development of an information technology artifact. Based on to Hevner 2004 study, this research should create an "object with an embedded solution to an understood research problem" [5] through the following process:

- **Problem Identification and Motivation:** "Define the specific research problem and justify the value of a solution" [5];
- **Define the objectives for a solution:** "Infer the objectives of a solution from the problem definition and knowledge of what is possible and feasible" [5];
- **Design and Development:** "Create the artifact" [5]. This can be "potentially constructs, models, methods, or instantiations" [5];
- **Demonstration:** "Demonstrate the use of the artifact to solve one or more instances of the problem" [5];
- **Evaluation:** "Observe and measure how well the artifact supports a solution to the problem. This activity involves comparing the objectives of a solution to actual observed results from use of the artifact in the demonstration" [5];
- **Communication:** "Communicate the problem and its importance, the artifact, its utility and novelty, the rigor of its design, and its effectiveness to researchers and other relevant audiences such as practicing professionals, when appropriate." [5].

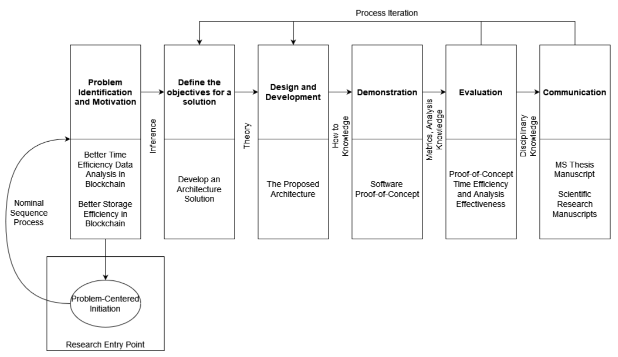

Process Iteration
Problem Identification and Motivation
Define the objectives for a solution
Design and Development
Demonstration
Evaluation
Communication
Inference
Theory
How to Knowledge
Metrics, Analysis Knowledge
Disciplinary Knowledge
Better Time Efficiency Data Analysis in Blockchain
Better Storage Efficiency in Blockchain
Develop an Architecture Solution
The Proposed Architecture
Software Proof-of-Concept
Proof-of-Concept Time Efficiency and Analysis Effectiveness
MS Thesis Manuscript
Scientific Research Manuscripts
Nominal Sequence Process
Problem-Centered Initiation
Research Entry Point

Fig. 1: Design Science Research Methodology

3 Data Analysis in Blockchain Distributed Systems: SLR

In this section, the systematic literature review is done in order to get a better understanding of the current research on data analysis in blockchain distributed file systems. With the acquired knowledge, the results are discussed and the artifact of this research is produced. In the DSRM, this section is responsible for the solution objectives definition.

3.1 SLR Planning

This section presents our three research questions. The three main topics these questions pretend to explore are blockchain, distributed file systems and data analysis, more specifically using streaming data techniques.

- **Research Question 1:** Which distributed file systems are used with blockchain?
- **Research Question 2:** How is data accessed and analyzed on architectures using blockchain and distributed file systems?
- **Research Question 3:** Which are the current streaming data architectures used in blockchain?

We used the search engine EBSCO Discovering Service [6] that includes the main research sources, such as Scopus, Academic Search and Clarivate Analytics (itself including Web of Science, Current Contents Connect, Derwent Innovations Index, MEDLINE e SciELO Citation Index, and other resources, such as Citation Reports and Essential Science Indicators).

To identify the relevant work, we used the following search expressions: (1) "AB (Blockchain) AND AB ("Distributed File System" OR "Decentralized File System" OR "Interplanetary File System")"; (2) "AB (Blockchain) AND AB ("Data Stream" OR "Data Streaming" OR "Data Flow" OR "Data Flows")".

The keyword AB indicates to the search engine we have used – EBSCO Discovery Service – that the search should be carried out in the title and the abstract. The papers were filtered automatically by the search engine according to Table 1.

The first search string resulted in 256 studies and the second in 111 studies. The merged results, after duplicates were removed, were 277 studies.

TABLE 1: Filtered Studies

Included	Excluded
Equivalent Subjects	Not Peer Reviewed
Full Text	Not Written in English
	Not Academic Journal or Conference Material

The studies abstracts were analyzed and classified as out of scope according to our inclusion/exclusion criteria, presented in Table 2.

The purpose of this criteria was to analyze novel data analysis architectures, such as new data access processes; new or different architectures for distributed file systems and blockchain or new distributed file systems technologies that were not included before. Studies with data management components were included since these could identify technical problems or solutions in current real world applications of these technologies.

An objective of this study is to understand how data analysis is being conducted in blockchain based systems, supported by distributed file systems. As such, blockchain specific technical improvements or blockchain technology integration in an industry such as using blockchain for agriculture, was deemed as out of scope. Personal data applications were likewise excluded since these are not in the scope of the study.

TABLE 2: Scope Inclusion/Exclusion Criteria.

Inclusion Criteria	Exclusion Criteria
Data Management	General Security Improvements
Data Processes	Personal Data Applications
Data Access Architectures	Specific Integration of Blockchain in an Industry
Different Distributed File Systems	Performance Improvements by Consensus Algorithms
Technologies Not included Before	

3.2 SLR Conducting

The abstract of every paper was studied which resulted in excluding a total of 181 papers on this phase. In the following phase we analyzed the introduction and conclusion of the remaining papers finishing this phase with a total of 30 papers.

In Figure 2, we can observe the distribution of the selected papers, where 2021 is the year with the most contributions, followed by 2019 and 2022. There were no limitations with regarding the date range of the papers selection. In the following subsections, the research results are divided by research question and the answers are presented by topic.

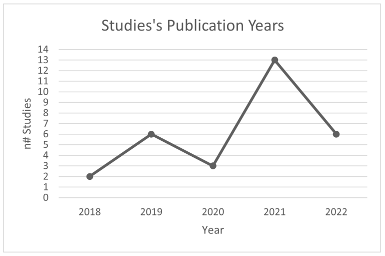


Fig. 2: Selected Studies Distribution by Year

3.3 SLR Reporting

3.3.1 RQ1: Which distributed file systems are used with blockchain?

Table 3 presents the distributed file systems in use as well as the blockchain being used when mentioned.

InterPlanetary File System (IPFS) is the most used distributed file system with blockchain in our sample. IPFS is a peer-to-peer hypermedia protocol where no nodes are privileged and a common computer system suffices as a node. The nodes store the IPFS objects in their local storage. Nodes then connect to each other and transfer objects. These objects represent the files and other data structures [7]. The object is chopped into smaller chunks of itself, hashed and given a unique content identifier (CID), which serves as a fingerprint. To access the object, the returned CID is necessary. IPFS "solve the shortage of blockchain in storing big files" [8] since "storing a document on the blockchain is expensive" [9].

Hadoop Distributed File System (HDFS) is the second most used distributed file system with blockchain in our studies sample. HDFS is an isolated master–slave data storage network composed of NameNodes and DataNodes. HDFS "is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets." [10]. HDFS is "mainly used for batch processing of data" [11]. HDFS is most suited when the nodes can be trusted.

Swarm is another distributed file system used with blockchain. Swarm is very similar to IPFS. Its biggest difference is that IPFS uses a Distributed Hash Table (DHT) and Swarm uses an immutable content address chunkstore to generate the content identifiers [12]. Swarm has a natural integration with Ethereum blockchain and an incentive system that benefits from smart contracts.

Merkle Tree based File System (MTFS) is a distributed file system that was integrated with blockchain. In MTFS a node consists of a "batch of servers with professional connection sitting in a data center" [13]. MTFS uses asymmetric cryptography including proxy re-encryption (PRE), to ensure data privacy. Its peer-to-peer network broadcasts data like a tree having redundant nodes and connections in case of failure. This file system has less adoption and implementation examples when compared with the previously mentioned file systems.

When adding data to a distributed file system, most of the studies follow a similar process, which can be summarized as follows:

1) **Data Source:** Create Data Entry and send to API
2) **API:** Send (Encrypted) Data to Distributed File System
3) **API:** Upload Data and Generate Hash from Data
4) **DFS:** Send Data's Hash to API
5) **API:** Send Transaction to Blockchain with the Data's Hash
6) **Blockchain:** Send Confirmation of Success to API

TABLE 3: Distributed File Systems Used

Distributed File Systems	Support Literature
IPFS and Ethereum	[8] [14] [15] [16] [9] [17] [18] [19] [20] [21] [22] [23]
IPFS and HLF	[24] [25] [26]
IPFS and Multi-Chains/ Custom-Chain	[27] [28] [29]
IPFS	[30] [31] [32] [3]
HDFS and Ethereum	[33]
HDFS and HLF	[11]
HDFS	[34]
MTFS	[13]
Swarm and Ethereum/ Hyperledger Fabric	[35]

3.3.2 *RQ2: How is data accessed for analysis on architectures using blockchain and distributed file systems?*

Table 4 presents the data access architectures used by blockchain and distributed file systems found.

Smart Contracts, or Custom Search Engine Query, are the most common data accessing mechanism among the distributed file system and blockchain architectures, within the research studies. In these methods, after the data content identifier is obtained from the distributed file system, the identifier is saved in the blockchain ledger, along with relevant metadata, such as access authorization. In the case of custom search engines it is also saved in a local or a cloud database. A smart contract or a traditional query in a local or a cloud database obtains the data content identifier by matching saved metadata such as a keyword. Using off-chain sources greatly improves access speed, however, since it is off-chain, it can be a target for malicious participants.

Hadoop Integration is the second most used accessing data mechanism identified. In these systems, the distributed file system used is HDFS where it is possible to use MapReduce that "is a pre-built framework in HDFS" [11]. In these cases, MapReduce can be used to analyze the data.

Share by Smart Contracts is another method used to access data from a distributed file system and a blockchain network where all the participants are trusted. The data content identifier is broadcast to all the participants through a smart contract. In this case every participant is able to directly access the saved file through the identifier in the distributed file system.

3.3.3 *RQ3: Which are the streaming data architectures used in blockchain?*

Only one streaming data architecture in blockchain was found in the analyzed studies - "ITrade: A Blockchain-based, Self-Sovereign, and Scalable Marketplace for IoT Data Streams" [37] (see Table 5). In this study, blockchain (Ethereum) and smart contracts are used for security, availability and trust purposes. Also, this system uses a pull-based message consumption model (Kafka) as the basis of its streaming architecture. This system's purpose is to give a data buyer the ability to subscribe to a data stream.

TABLE 4: Data accessed on Distributed File Systems and Blockchain

Data Access/ Analysis Found	Support Literature
Smart Contracts or Custom Search Engine Query	[8] [14] [15] [16] [9] [17] [30] [24] [33] [18] [19] [22] [20] [28] [26]
Hadoop Integration	[36] [11]
Shared by Smart Contract	[25]

TABLE 5: Streaming Architectures used in Blockchain

Data Streaming Architecture Found	Support Literature
Event-based Message Model	[37]

3.4 Discussion

Most blockchain architectures available in studies usually focus on adapting blockchain to an industry. In subsection 3.3.1, although different combinations of technologies are presented, (blockchains and DFS), the architecture between them is usually similar. Also, most of these architectures do not include or propose in their systems a mechanism or methodology for analyzing the data stored in their systems.

In subsection 3.3.2 we can observe the solutions used to access data. Most of them could be more efficient or secure making the analysis process under-performing. The smart contracts query system does not scale well and such these implementations are introduced with custom built search engines. The problem with custom build solutions is the lack of comparability across different frameworks. Also, since these solutions are not on-chain, they can be subject to malicious participants and do not work on a public blockchain. HDFS is naturally compatible with MapReduce. However, like the previously mentioned case, it is not suited for public settings, since HDFS intended use is when its nodes can be trusted. Likewise, the last solution found is also not suited for public settings. These results motivate the proposal of a different architecture.

4 Research Proposal

In the DSRM, this section presents the the design and development, the demonstration and the evaluation phases. The artifact of the design and development is the proposed architecture and the demonstration and evaluation are presented through the software proof-of-concept.

4.1 Design and Development

To improve the analysis process, we conceptualize an architecture that is divided into a data storage and collection layer composed by a distributed file system, integrated with blockchain based on the results analyzed in the systematic literature review and a data stream pipeline.

In Figure 3, we present an UML sequence diagram that showcases how new data is processed in the system. A user starts by sending data to the API through, for example, a website. The server's API can encrypt the data if needed and will send the data to a distributed file system to be saved. The distributed file system, after saving the data, will return the content identifier back to the API. The API will send a new transaction to the blockchain with the content identifier and if successful the confirmation of new data will be sent to both the API and then the user. After data is saved and the confirmation is sent to the user, the API will also send the new data to the data analysis pipeline for it to be readily available when an analysis request is submitted.

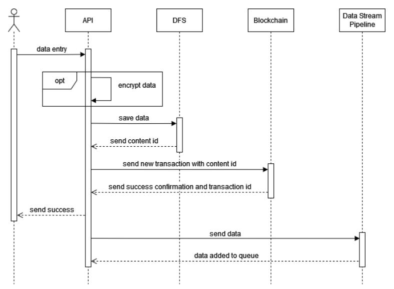


Fig. 3: Adding Data Process, UML Sequence Diagram

Figure 4 shows how data is accessed, as well as how an analysis request is fetched from the analysis results database. When a user sends a data request through, for example a website, a request is sent to the blockchain with the transaction identifier. Then, the blockchain returns the transaction data that contains the content identifier in the distributed file system. The content identifier is sent in a request to the distributed file system and the data is returned to the user by the API. The analysis request is sent to the data analysis pipeline and the requested analysis is returned from the data already analyzed in the database.

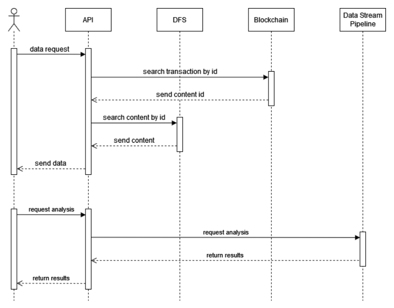


Fig. 4: Accessing Data Process, UML Sequence Diagram

The data stream pipeline is composed of an ingestion layer based on an event based message bus system (based on the results of subsection 3.3.3), a stream processing application and an incremental learning module. Incremental Learning is a machine learning method designed to ingest a continuous amount of data and continuously update the learnt model, which makes it ideal to a data stream. The model infers new statistical information based on new data; providing updated results while maintaining previously acquired knowledge.

In Figure 5 we can observe the analysis process inside the data stream pipeline. This pipeline is composed of three main parts. The data stream messages ingestion system is responsible for managing the incoming data to be analyzed from the API. The stream processing application requests the messages from the data stream messages ingestion system and processes the data and saves it, if necessary, in the results database, to visualize it. Lastly, the incremental learning algorithm pulls the data from the data stream messages ingestion system and the latest model from the database; it then processes the new data and updates the incremental learning model with the latest data. The stream processing application results may be of interest to the incremental learning algorithm and it is possible to use it as part of the input for the model. Using the stream processing application, efficient data pre-processing can be used.

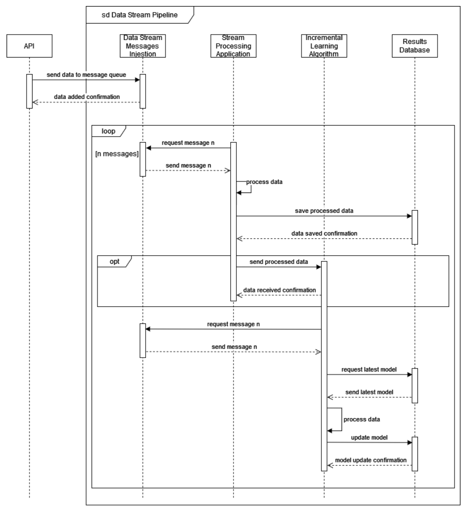


Fig. 5: Analyzing Data, UML Sequence Diagram

An identified challenge of typical distributed data stream processing frameworks is "how to accurately ingest and integrate data streams from various sources and locations into an analytics platform" [38]. Our proposed architecture solves this issue, since it aggregates multiple data sources into a single one through blockchain. It is also compatible with different types of blockchains (public or private), resulting in an architecture that is not bound to a single application.

Another issue solved by our proposed architecture is adding data analysis functionalities to an existing blockchain based system. For example, if there is a blockchain already being used, by applying our proposed pipeline and expanding the system's API, data analysis functionalities could be added.

One of the benefits of this new architecture is that it is modeled like microservices, since its modules are loosely-coupled as such, with small changes to the overall architecture, features can be added or removed (for example, encryption, access control or data pre-processing).

4.2 Demonstration

In this section, using the proposed architecture as the basis, a software proof-of-concept is implemented to showcase the DSRM's artifact usage. The technology used is presented and justified and the implementation environment described. Then an overview of the proof-of-concept and the development results are presented.

4.2.1 Proof-of-Concept Technologies

The chosen blockchain to create the proof-of-concept was Hyperledger Fabric (HLF). HLF is maintained by the Linux Foundation, an organization that supports open-source projects.

HLF is a permissioned distributed ledger framework that provides a foundation for developing applications or solutions with a modular architecture. The user uses the HLF client application to propose transactions on the network. The client can only read or write to the ledger, however, it is possible to delete data by sending a transaction that, in the application's logic deletes the said data. To propose transactions to the network, the client application needs to have a certificate from the Certificate Authority (CA). Chaincode are the name given to HLF smart contracts. Through container technology, chaincode implements the application's logic. In the proof-of-concept, the chaincode is implemented in Go Programming Language.

The HLF is composed by two types of node organizations. The orderer organisation responsible by the consensus algorithm of the blockchain and the peers organizations, responsible by committing the transactions to the orderer nodes, as well as keeping a copy of the ledger. In the proof-of-concept the consensus algorithm used is Raft. In the proof-of-concept CouchDB is used since it provides richer functionalities when compared with the alternatives.

There were four main reasons for choosing this blockchain framework. HLF "enables performance at scale while preserving privacy" [39] while still being highly customizable and modular. Secondly, it is one of the most used blockchain frameworks and is open-source, having multiple scientific studies using it in their research. Lastly, its only operation costs are the computation and storage of the servers used to execute it.

The distributed file system selected to integrate with HLF is the InterPlanetary File System IPFS. It is the most used distributed file system with blockchain in all the literature reviewed, as show in subsection 3.3.1. It is fast, scalable and allows any type of data to be stored. "IPFS is a distributed system for storing and accessing files, websites, applications, and data." [40]. In subsection 3.3.1, this technology usage explained.

The distributed event streaming platform chosen for the data stream pipeline that will manage the messages arrival for analysis is Apache Kafka. Kafka is composed of topics, producers, consumers and brokers. Kafka organization system works over topics. Every message that is sent to a Kafka system is sent to a topic. A topic is therefore a stream of messages. Each message (also named record), is stored in a key-value format. The key of this message is called Offset.

The producers are applications responsible for publishing messages to a given topic. Consumers are the applications that read said published messages from a given topic. Brokers are the instances responsible for exchanging messages with the producer and consumer applications. One broker is enough to implement a Kafka system, however, usually, multiple brokers are used to form a cluster and create replication. In the proof-of-concept a Kafka cluster is used with three brokers.

In a Kafka cluster, at the moment, it is necessary to implement a Zookeeper server with the purpose of managing and create consensus in said broker cluster. One Zookeeper server can be used to create said consensus but if needed, as long as in odd numbers, more Zookeeper servers can be added, creating a Zookeeper Cluster. In production, three or five Zookeeper servers are advised but in the proof-of-concept only one is used.

River library is the analysis tool selected to use Incremental Machine Learning (IML). "River is a Python library for online machine learning. It aims to be the most user-friendly library for doing machine learning on streaming data. River is the result of a merger between creme and scikit-multiflow." [41].

Since this is a proof-of-concept, stream processing capabilities were not implemented. The objective of these would be the transformation, cleaning and serializing of incoming data however this was not necessary since the test dataset used for evaluation was already prepared for analysis. If necessary, Kafka Streams integrates with Kafka and could be an option.

Lastly, all the application controllers and connectors are implemented using Python 3.8. All of the above mentioned technologies are free and open-source.

4.2.2 Proof-of-Concept Hardware Specifications

The development environment used to implement this proof-of-concept was a server with the following hardware specifications; CPU: Intel(R) Xeon(R) CPU E3-1240 v6 @ 3.70GHz; RAM:64GB; HDD:2TB; OS Version: Ubuntu Server 18.04. In this machine, with the use of virt-manager, a virtualization technology, twelve Virtual Machines (VM) were created with the following specifications: CPU:1 virtualCPU; RAM:4GB; HDD:30GB; OS Version:Ubuntu Desktop 20.04.4 LTS.

4.2.3 Proof-of-Concept Software Architecture

An internal network, with NAT, was also implemented for communication between said machines. This network was named Olympus with three organizations, Alpha, Beta and Omega.

Alpha organization is composed of four VMs, AlphaCA the Certificate Authority, AlphaAdmin the organization Admin and the two peers Zeus and Poseidon. Beta organization is composed of three VMs, BetaCA the Certificate Authority, BetaAdmin the Organization Admin and Hera the Organization Peer. Omega organization is composed of five VMs, OmegaCA the Certificate Authority, OmegaAdmin the Organization Admin and the three orderers Atlas, Cronus and Rhea. The Certificate Authorities and the Admin VMs are necessary for the setup of the HLF system however, when deployed, the network only needs the peers and the orderers for proper function.

The VMs used for the HLF peers are also used as peers for the IPFS cluster, Zeus, Poseidon and Hera.

In the data stream pipeline, BetaCA was used as a Kafka Broker, Hera was used as a Kafka Broker as well as a Producer and lastly, AlphaCA was used as a Kafka Peer, the Zookeeper Server, the Kafka Consumer and the IML computation module. Hera was chosen as the Kafka Producer since it is a HLF and an IPFS peer giving it access to the data.

The chosen VMs for Kafka could have been any others. For an easier implementation of adding data, one of the brokers chosen is a HLF and IPFS node. AlphaCA and BetaCA were chosen as the other brokers since, after setting up HLF they were not being used. Figure 6 shows the overral system at runtime.

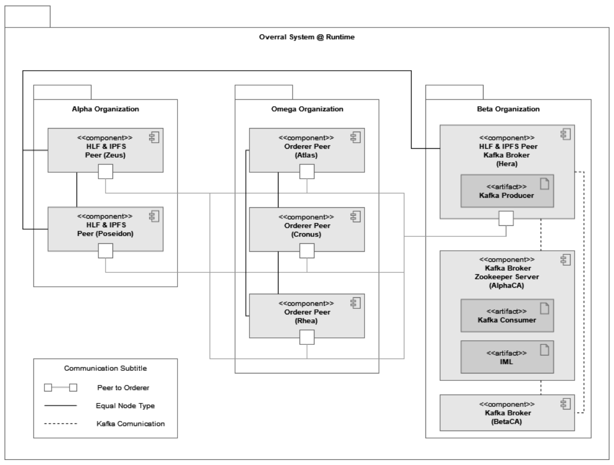

Overral System @ Runtime
Alpha Organization
<<component>> HLF & IPFS Peer (Zeus)
<<component>> HLF & IPFS Peer (Poseidon)
Omega Organization
<<component>> Orderer Peer (Atlas)
<<component>> Orderer Peer (Cronus)
<<component>> Orderer Peer (Rhea)
Beta Organization
<<component>> HLF & IPFS Peer Kafka Broker (Hera)
<<artifact>> Kafka Producer
<<component>> Kafka Broker Zookeeper Server (AlphaCA)
<<artifact>> Kafka Consumer
<<artifact>> IML
<<component>> Kafka Broker (BetaCA)
Communication Subtitle
Peer to Orderer
Equal Node Type
Kafka Comunication

Fig. 6: Overral System at Runtime, UML Component Diagram

4.2.4 Proof-of-Concept Implementation

The proof-of-concept code is available on GitHub [42]. The chaincode implemented in HLF, although not used in the proof-of-concept, allows the contents of IPFS, cid, to be deleted by changing the delete flag to True. It is also possible to return everything stored with GetAllAssets function. This chaincode available to be used by any peer of any peer organization.

Hera VM is responsible for adding, reading and deleting data from HLF and IPFS and sending all the added data to the chosen Kafka topic. The HLF, the IPFS and the Kafka Producer are controlled with bash commands implemented in python programs. All the bash commands are executed with the subprocess library. Hera's controller is responsible for integrating all these functionalities. AlphaCA VM is responsible for receiving and analysing data sent to a given Kafka topic. The analysis is made with the RiverML python library. AlphaCA's controller is responsible for integrating all these functionalities.

4.3 Evaluation

To perform evaluation in the proof-of-concept, the DSRM artifact demonstration, two data sources were used. The architecture and proof-of-concept, design allow for any kind of data to be analyzed. As such, the proposed architecture is cross-application and any data sources can be used.

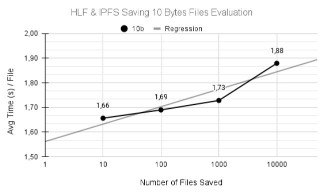
The first data source used were random files created with 10 bytes, 10 kilobytes, 1 megabyte using a python script. Through this source, the time it takes to add data to HLF & IPFS, and also, sending data and receiving data from the Kafka Cluster, is evaluated. The second data source is an dataset about the insurance costs of healthcare in the USA and was extracted from Kaggle [43]. With this source, the time it takes to save it in HLF & IPFS, send it to the Kafka Cluster, receiving data from said Cluster and analyze it with a simple Linear Regression is measured. The individual time of each component and the total amount (sum of all parts) is presented.

The test environment is the same as the demonstration implementation environment 4.2.2. To measure the time different components of the system take, the timeit python library is used.

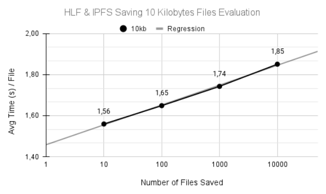
4.3.1 Produced Dataset

The Figures 7a and 7b represent the average time it takes to add 10, 100, 1000 and 10000 files with 10 bytes and 10 kilobytes to HLF and IPFS, respectively. Figure 7c represents the average time it takes to add 10, 100 and 1000 files with 1 megabyte of data (only evaluated until 1000 for lack of memory space) to HLF and IPFS. Hera VM was used.

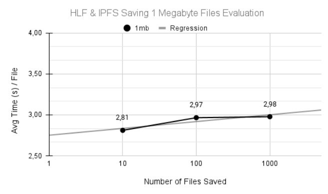
The total average time it takes to add a file to HLF and IPFS is 1.74 seconds for 10 bytes files, 1.70 seconds for 10 kilobytes files and 2.92 seconds for 1 megabyte files. The results show that the size of the file impacts the time it takes to add to the IPFS when files size are measured in megabytes. The results show that the time it takes to add files to HLF and IPFS scales linearly in terms of file quantity.



(a) HLF & IPFS 10 Bytes Files Evaluation



(b) HLF & IPFS 10 Kilobytes Files Evaluation



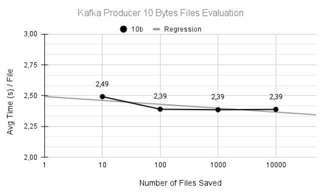
(c) HLF & IPFS 1 Megabyte Files Evaluation

Fig. 7: Save Files to HLF & IPFS Time Evaluation

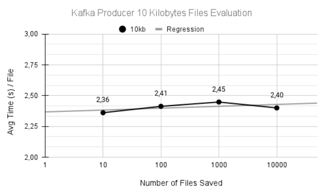
The Figures 8a and 8b represent the average time it takes to send 10, 100, 1000 and 10000 files with 10 bytes and 10 kilobytes to the Kafka cluster, respectively. Figure 8c represents the average time it takes send 10, 100 and 1000 files with 1 megabyte of data (only evaluated until 1000 for lack of memory space) to Kafka. Hera VM was used.

The total average time it takes to send a file to the Kafka cluster is 2.41 seconds for 10 bytes, 2.41 seconds for 10 kilobytes and 2.85 seconds for 1 megabyte. The results show that the size of the file does (or not) impact the time it takes to send data to the cluster. The results show that the time it takes to send these files to Kafka is mostly constant.

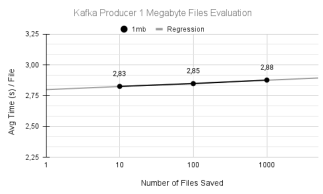
In total, saving a file in HLF & IPFS and sending its contents to the Kafka Cluster for analysis takes, on average, 4.15 seconds for 10 bytes, 4.11 seconds for 10 kilobytes and 5.77 seconds for 1 megabyte.



(a) Kafka Cluster Producer 10 Bytes Files Evaluation



(b) Kafka Cluster Producer 10 Kilobytes Files Evaluation



(c) Kafka Cluster Producer 1 Megabyte Files Evaluation

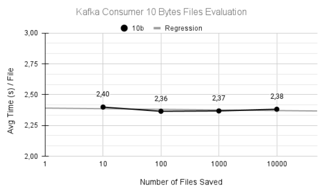
Fig. 8: Send Files to Kafka Cluster Time Evaluation

The Figures 9a and 9b represent the average time it takes to receive 10, 100, 1000 and 10000 files with 10 bytes and 10 kilobytes from the Kafka cluster, respectively. Figure 8c represents the average time it takes receive 10, 100 and 1000 files with 1 megabyte of data (only evaluated until 1000 for lack of memory space) from Kafka. AlphaCA VM was used. The times measured in the consumer are impacted producer since the evaluation was performed at the same time. If the data were sent to the topic and then consumed, the consumer time evaluation results would be better. However, this would not represent real world use cases, since the data production rate impacts the data consumption rate.

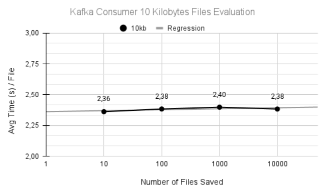
The total average time it takes to receive a file from the Kafka cluster is 2.38 seconds for 10 bytes, 2.38 seconds for 10 kilobytes and 2.42 seconds for 1 megabyte. The results show that the size of the file does (or not) impact the time it takes to receive data from the cluster. The results show that the time it takes to receive these files from Kafka is mostly constant.

4.3.2 Healthcare Dataset

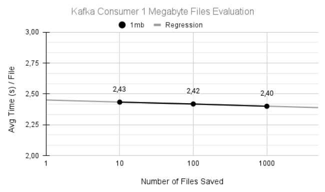
The healthcare insurance cost dataset [43] has 1338 entries. It has six columns, age, sex, body mass index, number of children in the insurance, smoker and region. The charges of the medical costs are the target of our analysis. The chosen algorithm to perform the analysis is a simple linear regression with a stochastic gradient descent of 0.05. The analysis metrics implemented are Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Coefficient of Determination (R2). These are common machine learning metrics for linear regressions. The model is saved every 50 data entries.



(a) Kafka Cluster Consumer 10 Bytes Files Evaluation



(b) Kafka Cluster Consumer 10 Kilobytes Files Evaluation



(c) Kafka Cluster Consumer 1 Megabyte Files Evaluation

Fig. 9: Receive Files from Kafka Cluster Time Evaluation

Using the above mentioned dataset, two approaches are used. The first scenario tests the time it takes to save, on HLF & IPFS, and send, with Kafka Producer, the single file with all the data, and the time it takes to receive it, with Kafka Consumer, and analyze it (building the model). In this approach, the time it takes on average, for data to be saved and sent is 3.31 seconds and is 51 minutes and 33.03 seconds to be analyzed. The second scenario is similar to the first but instead of using a single file, each data entry is separated through individual files. In this scenario, the time it takes on average for; data to be saved and sent is 1 hour 28minutes and 1.85 seconds; data to be analyzed 1 hour 27 minutes and 54.73 seconds seconds.

In both of the above mentioned scenarios, five random entries are sent to a different Kafka Topic, and used to perform predictions. On average, making a prediction for these five entries takes 11.17 seconds, not taking into account the time it takes to be sent from the producer. In these, the analysis metrics gave the following results: MAE: 2782.667; RMSE: 4051.730; R2: 0.888. These metrics can be improved using a more advanced algorithm, better data pre-processing, etc. Since these scenarios goal is to measure the duration of the analysis, (and showcase the architecture capabilities), the analysis results (metrics) were not optimized.

4.4 Discussion

In this section, an architecture was proposed with the goal of improving the time it takes to perform data analysis in blockchain systems integrated with DFS. By integrating blockchain and DFS it was possible to reduce the blockchain storage costs as well as improve its scalability and data types compatible.

Instead of making queries in the blockchain, in this architecture, data is stored in the system and sent to an data analysis module. Storing the data in the blockchain allows for data integrity. It also allows, depending on the implementation, to share data across organizations. If necessary, reading and extracting data from the blockchain is still possible using smart contracts. However, in this architecture, after confirming data has been stored with success, the data is sent for analysis.

To analyze data, IML is the chosen technology and has two main advantages, from an application perspective. First, it allows a model to be continuously improved and adapt to new data. Second, it allows a prediction to be made without stopping the training process. However, IML has two limitations. First, like machine learning, training takes time. Since data can be constantly added to the system, and sent for analysis, a ingestion system to manage the stream of data is necessary. Second, the input data, although with the use of smart contracts, completeness can be enforced, data preprocessing is limited. Using stream processing applications data can be pre-processed or visualized.

This architecture improves the time performance of the analysis, since a prediction can be requested to the system and is readily available. The main limitation is when the analysis intended is not built or trained. For example, when different IML algorithms need to be trained from the beginning and the data needs to be extracted from the blockchain (if the data is no longer available in the data stream messages ingestion system). For testing purposes, extracting the data to a data warehouse and using classic batch based machine learning may be faster. However, using IML, will provide better time efficiency when making up-to-date predictions.

In the software proof-of-concept, we can observe the average time it takes to save, access, send, receive and analyze data in a system using HLF, IPFS, Kafka, and RiverML. Through the evaluation, we can conclude it scales linearly. The time needed for saving the data and obtaining a model capable of producing a prediction is significant, however, optimizing the system with better distribution and improving the system hardware, in our case, VMs, will reduce the duration of the whole process. There were three main objectives achieved with this proof-of-concept. First, was demonstrate the architecture usage with existing technologies. Second, showcase how it would scale. Third, demonstrate, with a real-world dataset, the ability to make predictions from data while continuously improving a simple incremental machine learning model.

5 Conclusion

This technology stores data in architectures that provide high data integrity and provenance, as well as, a platform where different participants can share data with a high degree of trust. However, this data only has value if it can be accessed and analyzed in an efficient way creating, through the data analysis process, information. The use of a distributed file system can improve the amount, speed and type of data in blockchain. Also, streaming data technologies allow for a higher data flow from the moment data is accessed to the analysis.

The SLR was performed to identify which technologies were used with blockchain, the methodologies used to access data in these architectures and which streaming data architectures were being used.

Following the research results, an architecture is proposed based on the results from the SLR. The architecture is composed of blockchain technology, for trust, security, traceability, data integrity, data sharing and provenance purposes. A DFS is included in the architecture, for storage scalability and to store different data types such as files or images. Lastly, a data stream pipeline is included as a data analysis solution (with stream processing capabilities for data transformation on the go and/or incremental learning model(s) to analyze said data).

A software proof-of-concept is developed to demonstrate the use of said architecture without the stream processing module. A Hyperledger Fabric blockchain is deployed with the InterPlanetary File System as its DFS. Kafka is used as the distributed event streaming platform that controls the data flow and a Python library named RiverML is utilized as a incremental machine learning tool.

The proof-of-concept is evaluated using two a produced dataset and a healthcare dataset. Through the produced dataset, using files with different data sizes and different file quantities the proof-of-concept components are evaluated using time as the main metric. Through the produced dataset, we demonstrate the linear scalability of the system. Through the healthcare dataset, the architecture usage is exemplified with a real-world case.

5.1 Communication

This study and a scientific manuscript previously produced, "Benefits, Challenges and Solutions for Blockchain Data Analysis: SLR" submitted to an academic journal represent the communication phase of the DSRM. Through the SLR, the produced artefact (architecture) and the proof-of-concept prototype demonstration and evaluation, a novel data analysis architecture for blockchain and DFS is proposed to the community creating guidelines for future system architectures.

5.2 Research Limitations

This research is based on scientific literature only. However, the distributed file system, the blockchain and the data analysis topics also have developments described in gray literature. A multivocal literature review could be used to include that data, but that was not in the scope of our study.

5.3 Future Work

In this work, a novel architecture was produced. In order to showcase its use a software proof-of-concept was developed and evaluated. However, since the its purpose was showcasing the architecture only simple evaluations were done. Important work that could be done to further develop research in this topic include: Implementing and testing different technologies (blockchain frameworks, distributed file systems, incremental machine learning libraries and data streaming (and stream processing) platforms); Develop more an optimized version of the proof-of-concept, a prototype; Evaluating said prototype with, for example, load testing, spike testing, stress testing, volume testing, endurance testing and more scalability testing; Comparing system performance and analysis quality against a system that would send the data to a data warehouse and perform data analysis with classic batch based machine learning.

References

[1] D. D. Shin, "Blockchain: The emerging technology of digital trust," *Telematics and informatics*, vol. 45, p. 101278, 2019.

[2] W. Hou, B. Cui, and R. Li, "A survey on blockchain data analysis," *Proceedings - 2021 IEEE 45th Annual Computers, Software, and Applications Conference, COMPSAC 2021*, pp. 357–365, 7 2021.

[3] E. Daniel and F. Tschorsch, "Ipfs and friends: A qualitative comparison of next generation peer-to-peer data networks," *IEEE Communications Surveys and Tutorials*, vol. 24, pp. 31–52, 2022.

[4] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, no. 2004, pp. 1–26, 2004.

[5] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, vol. 24, pp. 45–77, 2007.

[6] I. S. T. (IST), "Ist digital library," https://bist.tecnico.ulisboa.pt/pesquisa/biblioteca-digital/, 2022, [Online; accessed 20-June-2022].

[7] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[8] W. Huang, "A blockchain-based framework for secure log storage," *2019 IEEE 2nd International Conference on Computer and Communication Engineering Technology-CCET*, 2019.

[9] K. Y. Bandara and J. Breslin, "Baas architecture for dapps and application for veterinary medicine case study in ireland," *The 2021 International Symposium on Networks, Computers and Communications (ISNCC 2021) - Dubai, UAE*, 2021.

[10] D. Borthakur *et al.*, "Hdfs architecture guide," *Hadoop apache project*, vol. 53, no. 1-13, p. 2, 2008.

[11] V. Mothukuri, S. S. Cheerla, R. M. Parizi, Q. Zhang, and K.-K. R. Choo, "Blockhdfs: Blockchain-integrated hadoop distributed file system for secure provenance traceability," *Blockchain: Research and Applications*, vol. 2, p. 100032, 12 2021.

[12] S. Team, "Swarm; storage and communication infrastructure for a self-sovereign digital society," https://www.ethswarm.org/swarm-whitepaper.pdf, 2021, [Online; accessed 27-June-2022].

[13] J. Kan and K. S. Kim, "Mtfs: Merkle-tree-based file system," *ICBC 2019 - IEEE International Conference on Blockchain and Cryptocurrency*, pp. 43–47, 5 2019.

[14] D. D. Taralunga and B. C. Florea, "A blockchain-enabled framework for mhealth systems," *Sensors*, vol. 21, 4 2021.

[15] M. Naz, F. A. Al-zahrani, R. Khalid, N. Javaid, A. M. Qamar, M. K. Afzal, and M. Shafiq, "A secure data sharing platform using blockchain and interplanetary file system," *Sustainability (Switzerland)*, vol. 11, 12 2019.

[16] A. Kumari and S. Tanwar, "A reinforcement-learning-based secure demand response scheme for smart grid system," *IEEE Internet of Things Journal*, vol. 9, pp. 2180–2191, 2 2022.

[17] Y. E. Oktian, S. G. Lee, and B. G. Lee, "Blockchain-based continued integrity service for iot big data management: A comprehensive design," *Electronics (Switzerland)*, vol. 9, pp. 1–36, 9 2020.

[18] Z. Zhou, M. Wang, Z. Ni, Z. Xia, and B. B. Gupta, "Reliable and sustainable product evaluation management system based on blockchain," *IEEE Transactions on Engineering Management*, 2021.

[19] R. G. R and P. K. S, "Self-restrained energy grid with data analysis and blockchain techniques," *Energy Sources, Part A: Recovery, Utilization and Environmental Effects*, 2020.

[20] N. Alrebdi, A. Alabdulatif, C. Iwendi, and Z. Lian, "Svbe: searchable and verifiable blockchain-based electronic medical records system," *Scientific Reports*, vol. 12, 12 2022.

[21] H. R. Hasan, K. Salah, I. Yaqoob, R. Jayaraman, S. Pesic, and M. Omar, "Trustworthy iot data streaming using blockchain and ipfs," *IEEE Access*, vol. 10, pp. 17 707–17 721, 2022.

[22] S. Jiang, J. Liu, L. Wang, and S.-M. Yoo, "Verifiable search meets blockchain: A privacy-preserving framework for outsourced encrypted data," *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, 2019.

[23] A. Desai, P. Shah, and D. D. Ambawade, "Verifyb - students' record management and verification system," *Proceedings - International Conference on Communication, Information and Computing Technology, ICCICT 2021*, 2021.

[24] E. Nyaletey, R. M. Parizi, Q. Zhang, and K.-K. R. Choo, "Blockipfs - blockchain-enabled interplanetary file system for forensic and trusted data traceability," *Proceedings - 2019 2nd IEEE International Conference on Blockchain, Blockchain 2019*, pp. 18–25, 7 2019.

[25] X. Tao, M. Das, Y. Liu, and J. C. Cheng, "Distributed common data environment using blockchain and interplanetary file system for secure bim-based collaborative design," *Automation in Construction*, vol. 130, 10 2021.

[26] C. Chen, J. Yang, W. J. Tsaur, W. Weng, C. Wu, and X. Wei, "Enterprise data sharing with privacy-preserved based on hyperledger fabric blockchain in iiot's application," *Sensors*, vol. 22, 2 2022.

[27] C. Peng, Y. Yu, J. Zhao, and H. Yu, "Research on cross-chain communication based on decentralized identifier," *HotICN 2021 - 2021 4th International Conference on Hot Information-Centric Networking*, pp. 7–12, 2021.

[28] A. S. Yadav, N. Singh, and D. S. Kushwaha, "Sidechain: storage land registry data using blockchain improve performance of search records," *Cluster Computing*, vol. 25, pp. 1475–1495, 4 2022.

[29] J. S. Gazsi, S. Zafreen, G. G. Dagher, and M. Long, "Vault: A scalable blockchain-based protocol for secure data access and collaboration," *Proceedings - 2021 IEEE International Conference on Blockchain, Blockchain 2021*, pp. 376–381, 2021.

[30] M. Chen, T. Malook, A. U. Rehman, Y. Muhammad, M. D. Alshehri, A. Akbar, M. Bilal, and M. A. Khan, "Blockchain-enabled healthcare system for detection of diabetes," *Journal of Information Security and Applications*, vol. 58, 5 2021.

[31] P. Altmann, A. G. Abbasi, O. Schelen, K. Andersson, and M. Alizadeh, "Creating a traceable product story in manufacturing supply chains using ipfs," *2020 IEEE 19th International Symposium on Network Computing and Applications, NCA 2020*, 11 2020.

[32] P. A. Lobo and V. Sarasvathi, "Distributed file storage model using ipfs and blockchain," *2021 2nd Global Conference for Advancement in Technology, GCAT 2021*, 10 2021.

[33] T. Renner, J. Muller, and O. Kao, "Endolith: A blockchain-based framework to enhance data retention in cloud storages," *Proceedings - 26th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2018*, pp. 627–634, 6 2018.

[34] F. Ishengoma, "Nfc-blockchain based covid-19 immunity certificate: Proposed system and emerging issues," *Information Technology and Management Science*, vol. 24, pp. 26–32, 12 2021.

[35] P. Sylim, F. Liu, A. Marcelo, and P. Fontelo, "Blockchain technology for detecting falsified and substandard drugs in distribution: Pharmaceutical supply chain intervention," *JMIR Research Protocols*, vol. 7, 9 2018.

[36] M. Hena and N. Jeyanthi, "A three-tier authentication scheme for kerberized hadoop environment," *Cybernetics and Information Technologies*, vol. 21, pp. 119–136, 12 2021.

[37] S. R. Niya, D. Dordevic, and B. Stiller, "Itrade: A blockchain-based, self-sovereign, and scalable marketplace for iot data streams," *Proceedings of the IM 2021 : 2021 IFIP/IEEE International Symposium on Integrated Network Management : 17-21 May 2021, Bordeaux, France, virtual conference*, 2021.

[38] H. Isah, T. Abughofa, S. Mahfuz, D. Ajerla, F. Zulkernine, and S. Khan, "A survey of distributed data stream processing frameworks," *IEEE Access*, vol. 7, pp. 154 300–154 316, 2019.

[39] "Hyperledger fabric team," last accessed 16 October 2022. [Online]. Available: https://www.hyperledger.org/use/fabric

[40] "Ipfs docs, what is ipfs?" last accessed 16 October 2022. [Online]. Available: https://docs.ipfs.tech/concepts/what-is-ipfs/#decentralization

[41] "River ml team," last accessed 16 October 2022. [Online]. Available: https://riverml.xyz/

[42] M. Baptista, "Github thesis proof-of-concept code," 2022, [Online; accessed 27-October-2022]. [Online]. Available: https://github.com/miguelarhb/Thesis

[43] M. Choi, "Medical cost personal datasets," last accessed 16 October 2022. [Online]. Available: https://www.kaggle.com/datasets/mirichoi0218/insurance