Data Analysis in Blockchain

Miguel Afonso Rodrigues Hipólito Baptista

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Miguel Leitão Bignolas Mira da Silva
Prof. Paulo José Osório Rupino da Cunha

Examination Committee

Chairperson: Prof. Manuel Fernando Cabido Peres Lopes
Supervisor: Prof. Miguel Leitão Bignolas Mira da Silva
Member of the Committee: Prof. João Carlos Ferreira

November 2022

Acknowledgments

First of all, I would like to express my sincere gratitude to my advisors, Prof. Miguel Mira da Silva and Prof. Paulo Rupino da Cunha, for welcoming me as their student and for their dedicated guidance and support throughout this work. It was an hard and rewarding journey and that was made possible thanks to these two excellent professionals and people. A special thanks to Prof. Cláudia Antunes that gave me useful constructive criticism during the Project Thesis resulting in this final Dissertation. I would also like to thank INOV for the space and work conditions it provided.

I would also like to thank my friends for the endless support and encouragement throughout all these college years, in special André Lopes, Daniel Ramalho, José Bartissol, Joana Teodoro, Mariana Chinopa, Leandro Salgado, Pedro Barbosa and Diogo Fernandes. A special thanks to my friends at AEIST and CPLEIC, two student groups that made me who I am. Also, my thesis collegue that helped me with part of this work implementation, Ricardo Martins Gonçalves, thank you.

Lastly, I want to thank my family, especially my parents and my brother for always being by my side during the good and bad times. Also, my grandparents for all the support given for my education.

To each and every one of you – Thank you!

Abstract

Blockchain and Data Analysis are two topics with increasing studies, and both are being integrated for multiple applications. However, accessing data on a blockchain is not a process as straightforward as on a regular centralized data repository, like a database.

In this work, two systematic literature reviews (SLR) are performed and a novel architecture is proposed. The main conclusions of these reviews are, (1) blockchain's main benefit is creating trust, security and privacy in a digital environment to gather data from different sources; (2) blockchain's main challenge is the time necessary to access and analyze stored data, and lack of tools to do so; (3) using distributed file systems (DFS) can avoid high storage and computation costs; (4) the most common way of accessing data in blockchain, although sub-optimal, is smart contracts.

With the gathered knowledge, a novel architecture is developed and presented. Besides blockchain's and DFS 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 is evaluated, showcasing its scalability and potential applications. Laslty, we present the main contributions, related work, research limitations and future work.

Keywords

Blockchain; Information System Security; Data Analysis; Data Streams; Incremental Machine Learning; Distributed File System

Resumo

Blockchain e análise de dados são dois tópicos com cada vez mais estudos e ambos têem sido integrados em múltiplas aplicações. No entanto, aceder a informação na blockchain não é um processo tão simples como num sistema central clássico de dados, como, por exemplo, uma base de dados.

Neste trabalho, duas revisões sistemáticas de literatura são realizadas e uma nova arquitetura é proposta. As principais conclusões destas revisões são, (1) o principal beneficio da blockchain é criar confiança, segurança e privacidade durante o processo de recolha de dados em diferentes fontes num meio digital; (2) o principal desafio da blockchain é o tempo necessário e a falta de ferramentas para aceder, analizar e guardar dados; (3) usar sistemas distribuidos de ficheiros pode evitar custos de armazenamento e computação elevados na blockchain; (4) a maneira mais comum de aceder dados na blockchain, embora aquém do ideal, são smart contracts.

Com os conhecimentos adquiridos, uma nova arquitetura é desenvovida. Para além das capacidades obtidas da junção da blockchain e dos sistemas distribuidos de ficheiros , o principal beneficio desta arquitetura é a capacidade de fazer previsões rápidas e atualizadas através do uso de aprendizagem incremental.

Uma prova de conceito que demonstra o uso da arquiteura é implementado, utilizando Hyperledger Fabric (Blockchain), o InterPlanetary File System (Sistema Distribuido de Ficheiros), Kafka (Plataforma Distribuída de Eventos e de Transmissão de Dados) e RiverML (Aprendizagem Incremental). O sistema é avaliado, demonstrando ser escalável e potenciais aplicações. Por último, as pricipais contribuiçoes, trabalho relacionado, limitações da investigaçao e trabalho futuro são apresentados.

Palavras Chave

Blockchain; Cibersegurança de Sistemas de Informação; Análise de Dados; Fluxos de Dados; Aprendizagem Incremental; Sistema de Ficheiros Distribuido

Contents

List of Figures

List of Tables

Listings

Acronyms

AB	Abstract
AI	Artificial Intelligence
API	Application Programming Interface
BC	Blockchain
CA	Certificate Authority
CID	Content Identifier
DHT	Distributed Hash Table
DSRM	Design Science Research Methodology
DLT	Distributed Ledger Technology
DFS	Distributed File System
HLF	Hyperledger Fabric
HDFS	Hadoop Distributed File System
IML	Incremental Machine Learning
IPFS	InterPlanetary File System
IT	Information Technology
IoT	Internet of Things
ML	Machine Learning
MAE	Mean Absolute Error
MTFS	Merkle Tree based File System
MSP	Membership Service Provider
NAT	Network Address Translation
RMSE	Root Mean Squared Error
R2	Coefficient of determination

SLR Systematic Literature Review

UML Unified Modeling Language

VM Virtual Machine

1

Introduction

Contents

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].

1.1 Research Background

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, the consensus algorithm verifies, among the blockchain participants, its validity. If valid, a new block is added to the blockchain.

"Blockchain can be divided into three types according to read-write permissions and ownership: public, private and consortium chain." [2]. Public blockchains are owned by all the nodes, stakeholders devices, where each one can read and write information to the blockchain. However, this usually results in a lower number of transactions per second in the network, since the computational power, used in the consensus algorithm, is higher when compared to the other variants. In private and consortium networks, only the owners, organization(s), or user(s) able to participate in the network, can read or write to the blockchain. The main difference between private and consortium blockchains is that consortium blockchains are usually owned by multiple organizations. This results in higher transactions per second, since the consensus algorithm is usually less computational intensive due to higher trust between parties. It is common for all the participating nodes in private blockchains to be tied to known identities outside of the network.

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. Additionally, all participants have equal access conditions to the stored data.

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 System (DFS) 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]. BitTorrent inspired modern DFSs as the first mainstream peer-to-peer data network [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 [4]. 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 gather value for a different number of industries.

1.2 Research Problem

Blockchain and Data Analysis are topics of high interest, and both are being studied and integrated for multiple applications [5]. However, research combining them does not provide guidelines on how to access data on a blockchain. This process is not as straightforward as on traditional database. Blockchain does not have a built-in query system, so most solutions can be classified into one of two categories: emulating querying with smart contracts and custom search engines; or extracting the data to a traditional database and accessing it from there. However, both solutions have issues. Querying data through smart contracts has high costs and slow performance, and extracting data to an off-chain database loses the data integrity protections afforded by the blockchain. There is also a lack of tools or frameworks to analyse data. The few solutions found were created for particular use cases or industries. Lastly, with smart contracts and custom search engines, analysing data stored in a blockchain is a time-consuming process and due to the nature of batch learning, it is a process that is repeated multiple times.

1.3 Research Objective

The objective of this research is to review the developments in the field and create a framework that can serve as a starting point in the development of tools to analyse data stored in blockchains. To that end, we propose an architecture based on microservices; so that, with minor changes, it will be cross-application. Using distributed file systems, it is possible to reduce storage costs and identify data tampering since, a content identifier, a hash of the data is saved in the blockchain. Furthermore, machine learning is used to analyse large amounts of data. By creating a proof-of-concept with existing open-source technologies, we demonstrate the feasibility of this framework.

1.4 Dissertation Outline

In Chapter 2, the research methodologies are presented. Chapters 3 and 4 showcase the first research methodology usage, one in each chapter. In Chapter 3, the research questions, main benefits

(3.3.1), challenges (3.3.2) and possible solutions (3.3.3) for said challenges of performing data analysis in blockchain are answered. Chapter 4 answers three research questions: (4.3.1) Which distributed file systems are used with blockchain? (4.3.2) How is data accessed on architectures using blockchain and distributed file systems? (4.3.3) Which are the current streaming data architectures used in blockchain?

Chapter 5 presents the research results application with a novel architecture. To demonstrate its usage, a software proof-of-concept is developed and evaluated. Lastly, Chapter 6 makes the conclusion remarks and presents the main contributions, research limitations and future work of this study.

2

Research Methodology

Contents

In this chapter, the research methodologies chosen to conduct this dissertation 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" [6].

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 [6] as shown in Figure 2.1, 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.

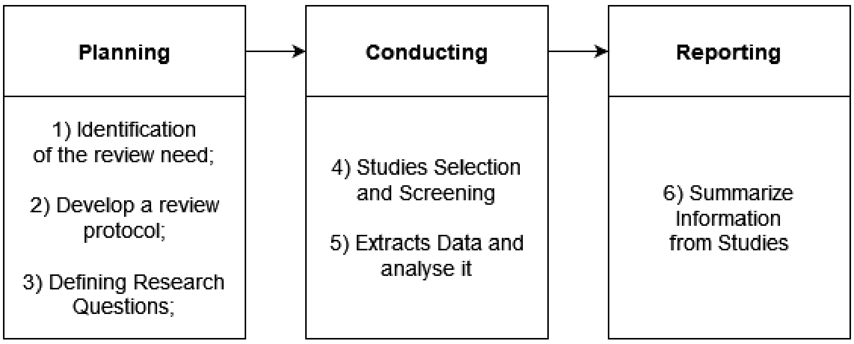

Figure 2.1: Systematic Literature Review Process

The SLR aim is to identify the problem and get answers to our proposed research questions.

2.2 Design Science Research

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 (IT) artifact. According to Hevner 2004 study [7], Design Science is adequate to solve problems that have,

- "unstable requirements and constraints based upon ill-defined environmental contexts complex interactions among sub-components of the problem and its solution" [7];

- “inherent flexibility to change design processes as well as design artifacts (i.e., malleable processes and artifacts)” [7];
- “a critical dependence upon human cognitive abilities (e.g., creativity) to produce effective solutions” [7];
- “a critical dependence upon human social abilities (e.g., teamwork) to produce effective solutions” [7].

The above reasons match with our research characteristics. As such, this research should create an “object with an embedded solution to an understood research problem” [8] through the following process:

Problem Identification and Motivation: “Define the specific research problem and justify the value of a solution” [8];

Define the objectives for a solution: “Infer the objectives of a solution from the problem definition and knowledge of what is possible and feasible” [8];

Design and Development: “Create the artifact” [8]. This can be “potentially constructs, models, methods, or instantiations” [8];

Demonstration: “Demonstrate the use of the artifact to solve one or more instances of the problem” [8];

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” [8];

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.” [8].

2.3 Research Outline

Figure 2.2 illustrates the DSRM applied to this research. The research problem and motivation are presented in Chapter 1. Then, two systematic literature reviews are conducted.

In Chapter 3 the first SLR is presented. Its goal was to define the specific research problem and justify the value of a solution. It also allowed to acquire knowledge of the status of this topic in the scientific body of knowledge.

The second SLR conducted in Chapter 4 allowed an understanding of what is possible and feasible. Through this research the objectives for the solution were defined, resulting in guidelines for the artifact.

In Chapter 5 a novel architecture is proposed to analyze data stored in blockchain and distributed file systems. In the DSRM this architecture represents the artifact design and development phase. The proof-of-concept is developed to demonstrate and evaluate the architecture usage.

The communication phase is achieved with this dissertation and the scientific manuscripts created. It is presented in the Chapter 6. This concludes the research in accordance with the DSRM.

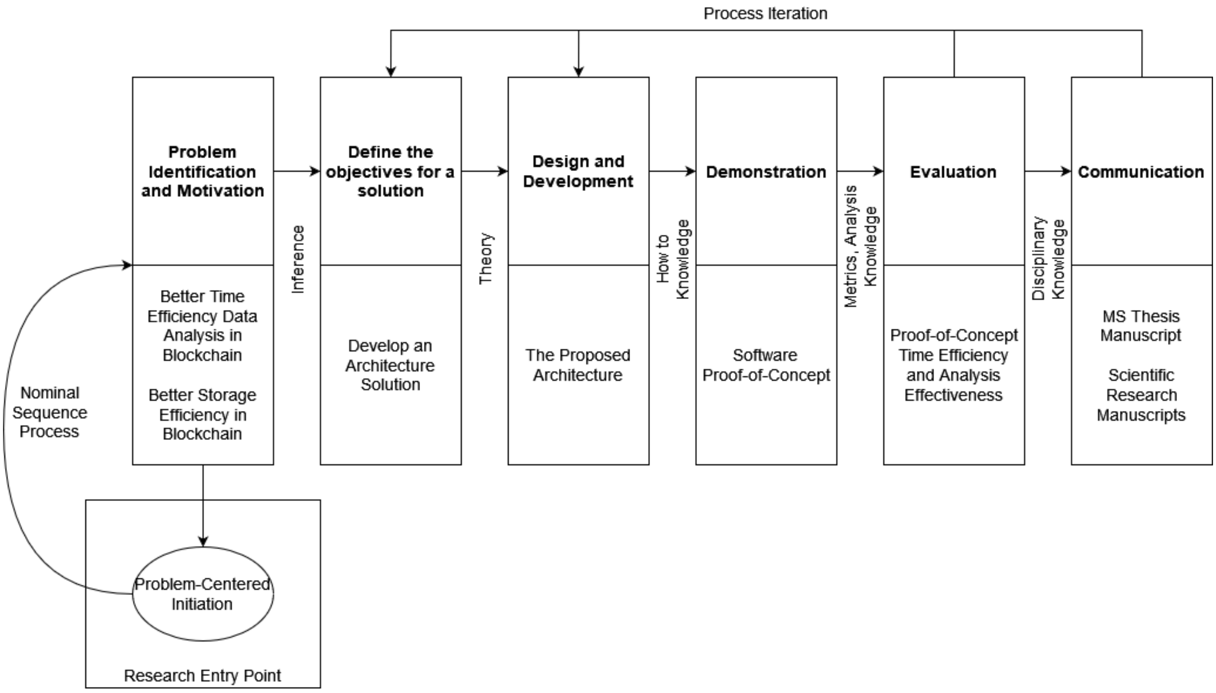

Figure 2.2: Design Science Research Methodology

3

Benefits, Challenges and Solutions for Data Analysis in Blockchain

Contents

In this chapter the systematic literature review is conducted and described. The purpose of this SLR is to identify current problems and justify the motivation of this research. Also, benefits and possible solutions are identified.

3.1 SLR Planning

This section presents our three research questions. The two main topics of these questions pretend to explore are blockchain and data analysis in blockchain. When performing data analysis on a Blockchain:

Research Question 1: What are the main benefits?

Research Question 2: What are the main challenges?

Research Question 3: What solutions can be used?

To identify relevant work, we used the following search expression: “Abstract (AB) (blockchain OR Distributed Ledger Technology (DLT)) AND AB (“data analysis” OR “data analytics” OR “business analytics” OR “data handling”)”.

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.

We used the search engine EBSCO Discovering Service that includes the main 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).

3.2 SLR Conducting

The criteria were applied on the search engine, filtering the studies automatically. Studies that were from equivalent subjects to the topics searched and had the full text available were included. Studies that were not peer reviewed, not written in English and were not academic journals or conference materials were excluded. The publication date was not considered an inclusion or exclusion criterion.

Based on the search results and the use of the above criteria, and after removing duplicates, we obtained 299 studies. The abstract of every paper was then analyzed, which resulted in excluding some studies for being out of scope (178) or having the wrong subject (12), leading to the removal of a total of 190 papers in this phase.

In the following phase, we analyzed the introduction and conclusion of the remaining papers, finishing this phase with a total of 41 papers. This process is represented in Figure 3.1.

In Figure 3.2, we can observe the distribution of the selected papers, where 2019 and 2020 are the years with the most contributions to this research, followed by 2021. From the total selection, 24 are

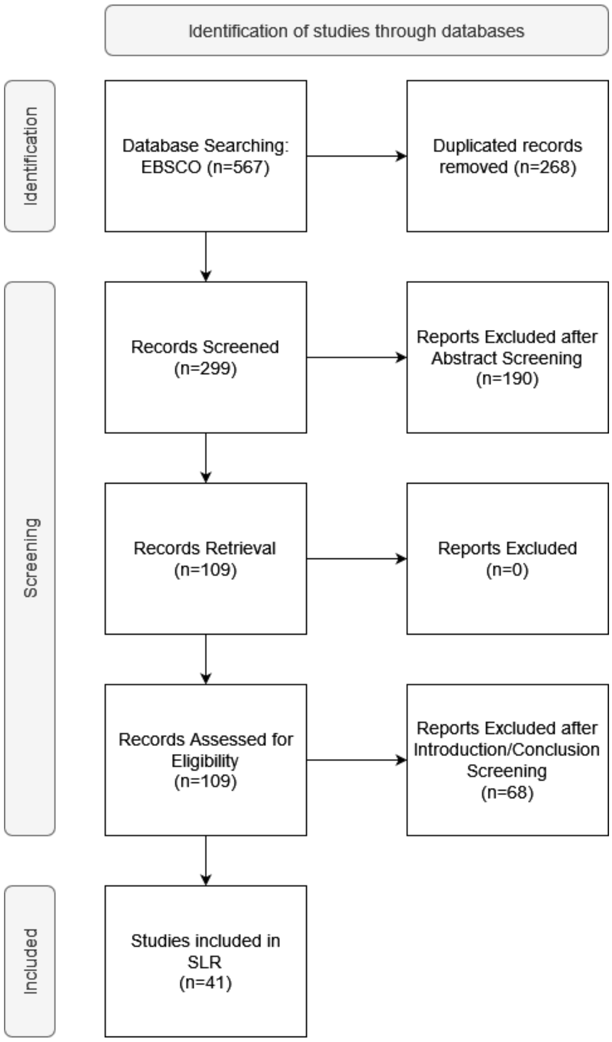

Figure 3.1: PRISMA flow diagram for the Systematic Literature Review

journal articles, and 17 are conference papers.

3.3 SLR Reporting

In this section, the SLR results are presented and organized in tables using the support literature to each answer found.

3.3.1 RQ1: What are the main benefits?

Table 3.1 presents the main benefits we identified and the respective supporting literature. The topmost benefit of the blockchain data analysis process is, according to the selected literature, the ability to store, collect and share data with and from different participants or sources. Smart contracts can have

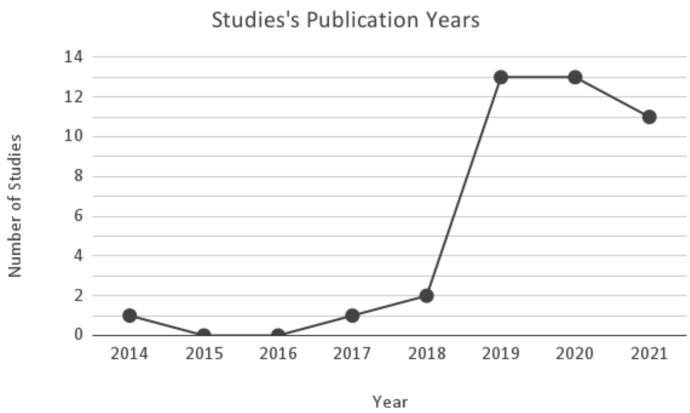

Figure 3.2: Selected Studies Distribution by Year

the capacity to enforce "decentralized access control for data sharing and analytics" [9]. This allows for the security and privacy of data. For example, an architecture where the stored information is encrypted and the data owner has the ability, through a smart contract, to share its key; allowing the owner to have an extra layer of security, since access to the key is necessary. This architecture would also allow better privacy, since the data is only shared with the intended participants. Also, with the option of implementing incentive-based systems, through cryptocurrencies, crowdsourcing environments can be produced. These environments can foster secure and fair data collection, improving the amount of data available for analysis.

Distributed and/or parallel data processing is blockchain's second most mentioned benefit. Blockchain is a distributed network of interconnected systems. However, this is not as fast as regular modern distributed clusters, mainly due to the processing needed to maintain network security.

In a blockchain network, the distributed ledger is available across multiple nodes. This enables high availability and consistency, and with blockchain's inherent secure architecture, data is extremely unlikely to be lost or corrupted. Also, smart contracts can enforce data structures when appending a block to the blockchain improving accuracy and completeness. This results in high data integrity, an important characteristic when dealing with data.

Blockchain metadata can benefit the blockchain data analysis phase since it can feed the machine learning models [10] and increase the overall quality of the decision/prediction result. For example, the number of transactions can correlate to past stored data trends, improving the model's prediction capabilities for the short term.

Lastly, data provenance is a "historical record of the data and its origins showing the trails of entities and processes that influenced data of interest" [9] made possible by blockchain distributed ledger immutability.

Table 3.1: Main Benefits Found

Main Benefits Found	Support Literature
Data Sharing/ Collecting/ Storage	[9], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20] [21], [22], [23], [24], [25]
Distributed and/or Parallel Data Processing	[9], [10], [15], [20], [22], [24], [26], [27], [28]
Data Integrity	[9], [15], [22], [29], [30]
Blockchain Metadata	[9], [10], [31], [32], [33]
Data Provenance	[9], [21], [34]

3.3.2 RQ2: What are the main challenges?

Table 3.2 summarizes the main challenges identified and the supporting literature for each challenge. The time it takes to access data significantly impacts the data analysis process. Currently, real-time data access is “essential as a blockchain is a highly dynamic system” [9]; however, the synchronous communications of the blockchain network make it difficult to do so. Also, querying data in the blockchain is time consuming because the data blocks “are written into files on disk” [35], and “Data storage models in blockchains are rather limited and optimized for storage, rather than for searching and indexing, unlike the conventional counterparts” [9].

System performance and storage waste are also commonly identified challenges to various blockchain-based systems. “The storage of irrelevant data wastes computational resources” [24].

The lack of analysis service integration in systems and the lack of maturity and research, between these two topics, were also pointed out in the studies. There is a need to develop custom-made solutions [18] for most blockchain data analysis contributing to an increase in the effort it takes to analyse the data.

Privacy problems such as being unable to maintain the privacy of sensitive data in the future (due, for example, to the computational evolution) is a data governance issue. Another matter mentioned in the literature derives from government policies such as General Data Protection Regulation in the European Union [36]. The immutability properties of blockchain do not allow data to be deleted, thus violating one of the articles of the regulation.

Lastly, data heterogeneity was identified as a challenge since the number of data sources (on-chain and off-chain) a blockchain is exposed, makes it harder to monitor and manage when compared to conventional systems [10, 22].

3.3.3 RQ3: What solutions can be used?

Table 3.3 shows solutions that can be used to tackle some of the main challenges identified previously and the supporting literature.

Table 3.2: Main Challenges Found

Main Challenges Found	Support Literature
Time to access data	[9], [33], [35], [37], [38]
System Performance and Storage Waste	[9], [24], [39], [40]
Lack of Research/Tools for Data Analysis and Blockchain	[18], [27], [33], [38]
Data Governance	[9], [26], [34]
Data Heterogeneity	[2], [22]

According to [2], “The number of papers reflect that machine learning has already become the mainstream of blockchain data analysis”. Machine learning is also used to improve the security of blockchain systems. Machine learning is an indispensable component of data analysis since it enables the analysis of blockchain's stored data and the decision/forecasting process automated from large data sources, such as a blockchain-based system. The most commonly found methodology to perform data analysis was to extract the data to a traditional information system and analyze it through machine learning models. Integration between novel machine learning methodologies, computing architectures, and blockchain is also being explored. One example is the growing field of blockchain networks integrated with federated learning for secure edge computing.

Hyperledger is mentioned as the basis of several architectures and business solutions. The studies presented contain technology improvements to this platform and new tools and integration. Smart Contracts are also presented as a solution because of the capacity to enforce data structures and data access.

There are different approaches taken to improve data access time. By skipping the querying process altogether, analyzing the data locally in each node, or optimizing the query process, it is possible to improve the time necessary to access and use data. Rahasack [37] is a custom enterprise-focused solution that analyses data in the blockchain by integrating it with a data streaming pipeline.

Blockchain's decentralization allows using its computing resources in a way that was not previously available in such a secure form. Such is the case for Edge Computing, Grid Computing, Cloud Computing, Ad-Hoc Computing, and some novel hybrid variations that allow a distributed data analysis process. Stream Computing was also proposed to analyze data in real-time [26]. Furthermore, Hadoop MapReduce is also proposed for blockchain-based systems due to its ability of parallel processing and compatibility with other data analysis tools. Lastly, there is a study that showcases data evaluation tools [46].

Table 3.3: Solutions Found

Main Solutions Found	Support Literature
Machine Learning	[2], [10], [14], [16], [24], [28], [41], [42], [43], [44]
Hyperledger Fabric	[13], [20], [21], [23], [31], [45]
Smart Contracts	[9], [11], [19], [25], [43], [46]
Skip Loading/ Querying Data Process	[22], [27], [29], [47]
Optimized Data Access/ Query Process	[12], [37], [48]
Hadoop MapReduce	[20], [22], [49]
Edge Computing	[20], [28], [39]
Cloud Computing	[22], [26]
Hybrid Variations	[20], [22]
Ad-Hoc Computing	[19]
Grid Computing	[22]
Stream Computing	[26]
Data Evaluation Tools	[46]

3.4 Discussion

Blockchain technology is being used in many diverse fields, such as supply chain, smart ecosystems, industry 4.0, crowdsourcing, education, and others, due to its ability to provide a secure, distributed, immutable ledger that creates trust between participants. The trust, security, privacy and anonymity that blockchain architectures can motivate cooperation between network participants, by sharing their data. Sharing data can be valuable for businesses that interact with each other or their customers.

The blockchain's ledger decentralization allows for increased data integrity, high availability, since there are multiple machines with no single point of failure, and high consistency, since no data is changed or deleted and is equal in every version of the ledger because of the consensus algorithm. Furthermore, with smart contracts, it is also possible to enforce the data structure and access.

The abovementioned characteristics create a high-value data environment where data can be gathered and stored. In addition, this process also generates metadata that can further increase the amount of data fed to the data analysis models and thus increase the quality of the end results.

Blockchain compatibility with modern computing solutions is also of great value since, nowadays, the amount of data stored and analyzed is increasing exponentially. While reviewing the studies, there was an increasing trend of off-loading the computation necessary to perform analysis to the nodes.

The distributed nature of blockchain also creates difficulties. Since it works with synchronous communications, access time can be time-consuming.

Some solutions were briefly mentioned in subsection 3.3.3 as being the most commonly used, resort to off-loading the data to a traditional system to be analyzed conventionally. However, this solution creates a problem. The data stored in the traditional system can be tampered decreasing the trust in this solution.

Another solution was to improve the query system. One proposed way was adding headers to the blockchain's blocks and query using those headers. However, this can become inefficient over time.

Data streaming architectures were also proposed to tackle the time it takes to access data. These avoid off-chain tampering and scale well over time, but the analyzed solution was custom made, lacking flexibility for the multiple blockchain applications and creating a high amount of development effort.

Machine learning techniques were found to be used to improve blockchain vulnerabilities or performance problems. However, for the analysis of stored data in the blockchain, the most commonly found research was federated learning. This Machine Learning (ML) technique enables distributed training of models without having access to the private information by training said model in the nodes and then aggregating the various models into one. With blockchain, this aggregation can be decentralized, accomplishing better security. Also, the data provenance stored in a blockchain can help achieve the reproducibility of a data analysis model through its immutability properties.

The lack of tools/architectures for blockchain data analysis and the lack of integration with conventional tools due to the novelty of blockchain technology was an identified issue. "Research efforts that examine both Big Data and blockchains topics, either describe general capabilities of the two emerging technologies or examine cases that are not focused on the analysis of Big Data but in other procedures like authentication and access control." [27]. Most of the literature found during this research, explored the application of this technology in different fields, but very few researched the data analysis process on the blockchain.

Data governance issues are being researched. Data privacy issues introduced by blockchain architecture still have topics for debate. For example, the ability to maintain the privacy of sensitive data in a public blockchain. With the increase of computational power, in the future, sensitive data may be compromised, since the encryption that makes the information private [9] may be subjected to brute force attacks. The most common proposed solution for these issues, apart from storing data in a similar architecture to edge computing, is to introduce an off-chain. Off-chain data storing has some advantages, such as cheaper storage [50]; however, this introduces issues such as assuring data immutability and increased difficulty for analysis services integration.

4

Data Analysis in Blockchain Distributed File Systems

Contents

In this chapter, 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 Chapter is responsible for the solution objectives definition.

4.1 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 [51] 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")".

4.2 SLR Conducting

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 4.1.

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 4.1: 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 4.2.

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 4.2: 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	

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. Figure 4.1 represents the process through a PRISMA flow diagram.

In Figure 4.2, 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.

4.3 SLR Reporting

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

Table 4.3 presents the distributed file systems in use as well as the blockchain being used when mentioned.

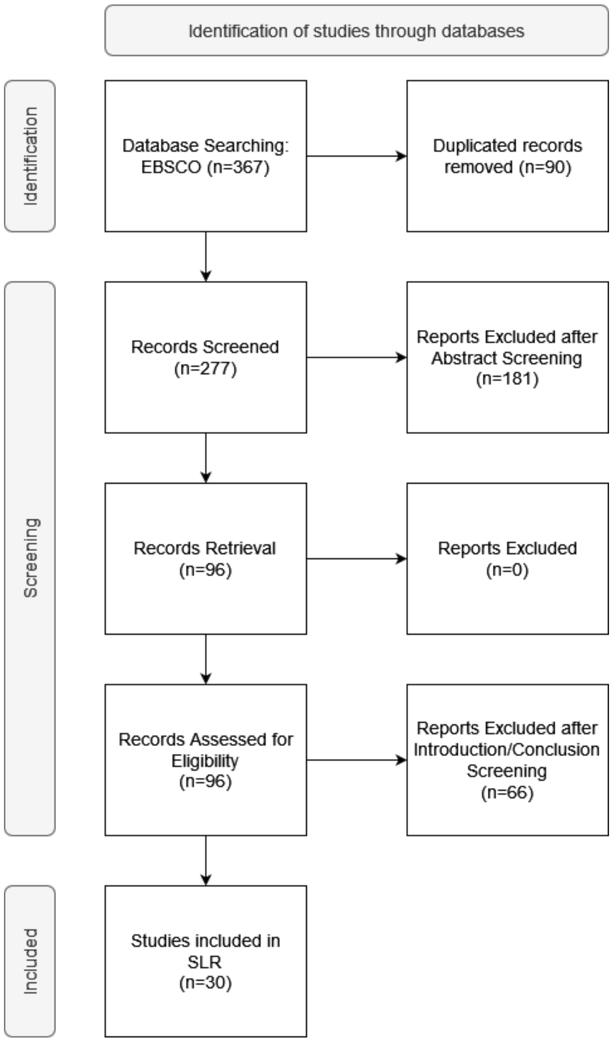

Figure 4.1: PRISMA flow diagram for the Systematic Literature Review

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 [52]. 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" [53] since "storing a document on the blockchain is expensive" [54].

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 hard-

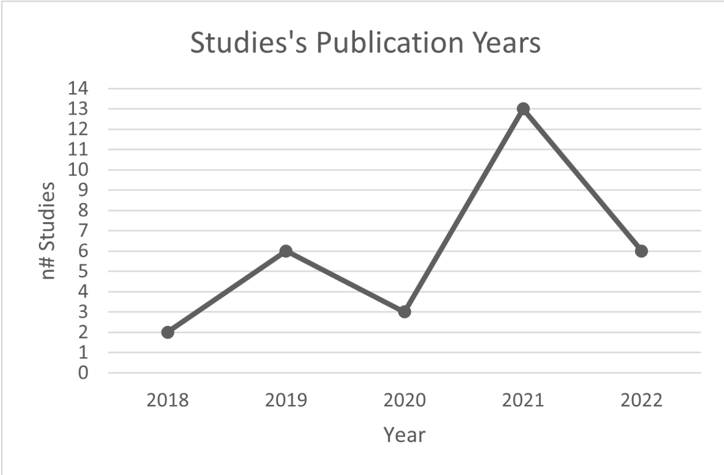

Figure 4.2: Selected Studies Distribution by Year

ware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets." [55]. HDFS is "mainly used for batch processing of data" [56]. 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 [57]. 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" [58]. 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 Application Programming Interface (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 (BC):** Send Confirmation of Success to API

Table 4.3: Distributed File Systems Used

Distributed File Systems	Support Literature
IPFS and Ethereum	[53] [59] [60] [61] [54] [62] [63] [64] [65] [66] [67] [68]
IPFS and Hyperledger Fabric (HLF)	[69] [70] [71]
IPFS and Multi-Chains/ Custom-Chain	[72] [73] [74]
IPFS	[10] [75] [76] [3]
HDFS and Ethereum	[77]
HDFS and HLF	[56]
HDFS	[78]
MTFS	[58]
Swarm and Ethereum/ Hyperledger Fabric	[79]

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

Table 4.4 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" [56]. 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.

Table 4.4: Data accessed on Distributed File Systems and Blockchain

Data Access/ Analysis Found	Support Literature
Smart Contracts or Custom Search Engine Query	[53] [59] [60] [61] [54] [62] [10] [69] [77] [63] [64] [67] [65] [73] [71]
Hadoop Integration	[80] [56]
Shared by Smart Contract	[70]

4.3.3 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" [81] (see Table 4.5). 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 4.5: Streaming Architectures used in Blockchain

Data Streaming Architecture Found	Support Literature
Event-based Message Model	[81]

4.4 Discussion

Most blockchain architectures available in studies usually focus on adapting blockchain to an industry. In subsection 4.3.1, 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 4.3.2 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.

4.5 Review's Related Work

There are various SLRs reviews in the research field of blockchain and its applications on industries in general or in specific applications like healthcare [82], supply chains [83], energy [84], Internet of Things (IoT) [85] and smart cities [86], finance [87], government [88], education [89], agriculture [90], etc. There are also various reviews in the research field of blockchain and different technology improvements to blockchain security [91] and privacy [92].

Also, H. Huang [93] makes a summary of the current state of blockchain and DFS, demonstrating challenges and open issues. However, this study is not a SLR and N. Deepa [94] presents a survey about the state of big data and blockchain, including the data analysis topic; however, it only mentions distributed file systems briefly. As such, while researching this field, no SLR was found that addresses data analysis in blockchain (and DFS) with the same scope. Also, Adedoyin A. Hussain [95] is not an SLR and focuses on Artificial Intelligence (AI) and blockchain integration without considering every phase of the analysis process.

5

Research Proposal

Contents

In the DSRM, this chapter 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.

5.1 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 5.1, we present an Unified Modeling Language (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.

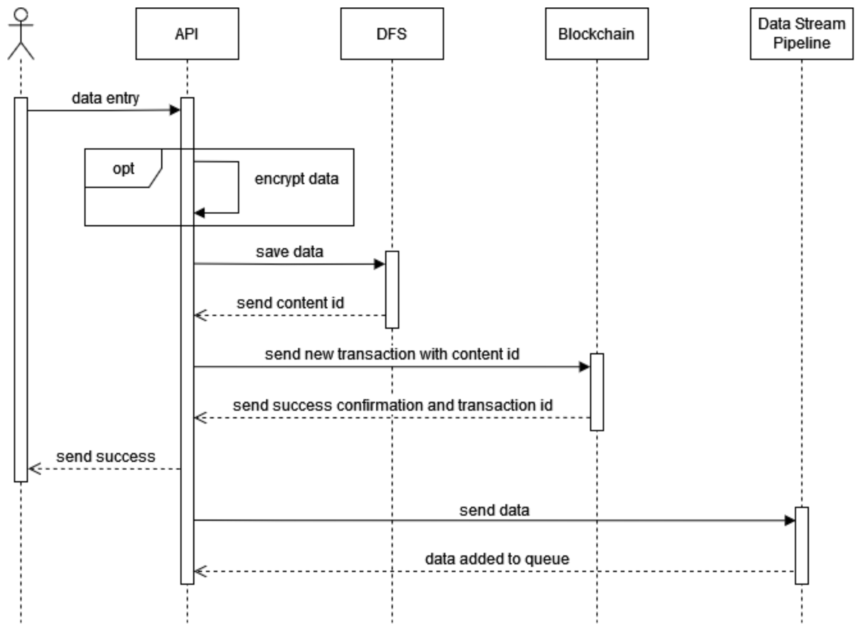

Figure 5.1: Adding Data Process, UML Sequence Diagram

Figure 5.2 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.

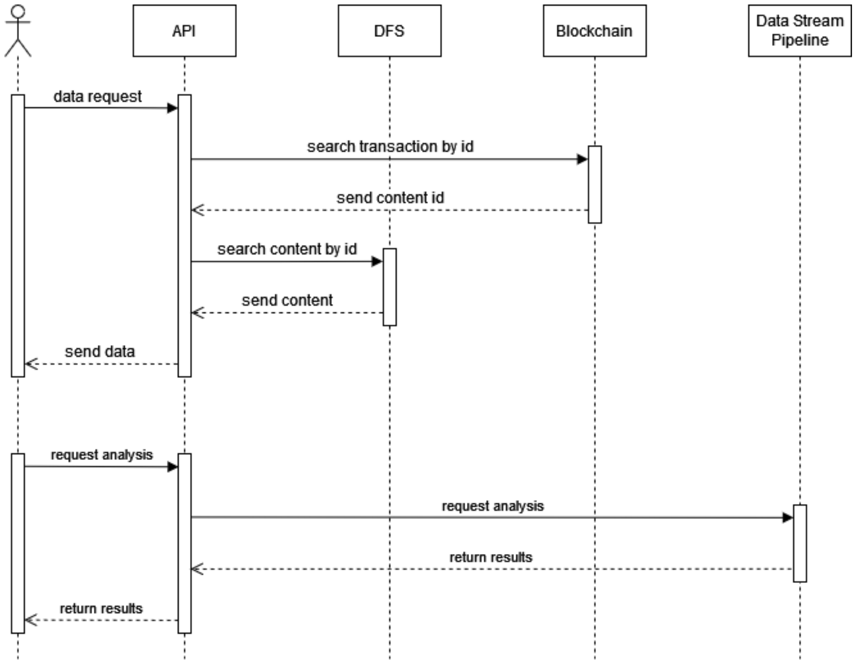

Figure 5.2: Accessing Data Process, UML Sequence Diagram

The data stream pipeline has three components: the ingestion layer, event-based message bus system (based on the results of subsection 4.3.3); a stream processing application; 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 using the new data; providing updated results while maintaining previously acquired knowledge [96].

In Figure 5.3, we can observe the analysis process inside the data stream pipeline. 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. Lastly, the incremental learning algorithm pulls the data from the data stream messages ingestion system and the latest model from the database; then, it 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. With a stream processing application, efficient data pre-processing can be integrated.

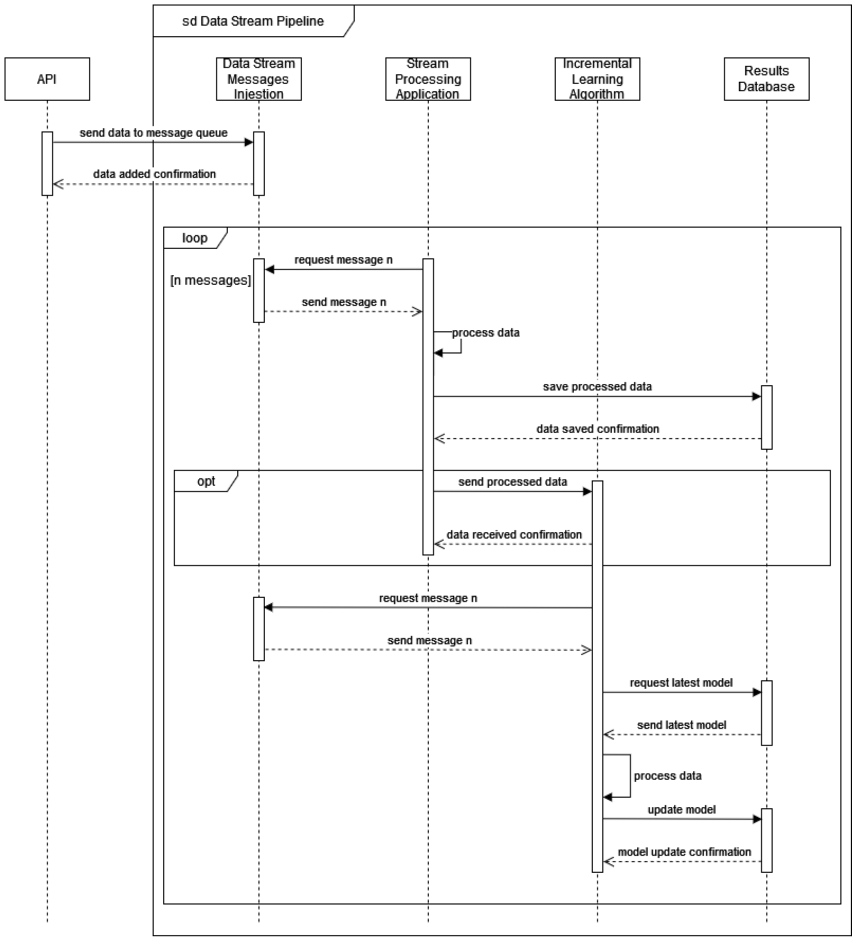

Figure 5.3: Analyzing Data, UML Sequence Diagram

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" [97]. 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: applying the proposed pipeline and expanding the system's API on a blockchain already in use, would add data analysis functionalities.

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

5.2 Demonstration

In this section, using the proposed architecture as the basis, a software proof-of-concept implements the DSRM's artifact usage. The chosen technologies and implementation environment are presented, as well as, the proof-of-concept overview.

5.2.1 Proof-of-Concept Technologies

The chosen blockchain to create the proof-of-concept was Hyperledger Fabric. HLF is maintained by the Linux Foundation [98], 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. HLF blockchain framework has four main components: the user (client), the nodes (peer or orderer), and the Membership Service Provider (MSP).

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).

The MSP defines HLF rules. It is responsible for validating, authenticating, and allowing access to every identity that participates in the network. MSP, through the use of a CA, allows and revoke the user's certificates. There are two types of MSPs: the local MSP and the channel MSP. The local MSP defines the users and nodes and who has administrative or participant rights. The channel MSP defines who has administrative or participant rights in a channel.

Channels are restricted messaging paths used by HLF to provide transaction privacy and confidentiality. All the data on a channel is therefore invisible and inaccessible to channel outsiders. The data includes the chaincode, transactions, members, and channel information.

Chaincode is 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 of two types of node organizations: the orderer organization and the peer organization(s). The orderer organisation is responsible for the consensus algorithm of the blockchain. The peer organization(s) is responsible for 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 [99].

The ledger has two different parts, the world state and the blockchain. The blockchain records all the changes that occur in the world state. The world state is a database that holds the ledger current state, expressed in key-value pairs. The world state is only maintained by the peers. In the proof-of-concept CouchDB is used since it provides richer functionalities when compared with the alternative, LevelDB.

The blocks of the blockchain contain the header (block number, current block hash, and previous block hash), the data, the metadata (time, certificate, public key, signature, and validity), and the transaction. The transaction is composed of the transaction header (with transaction metadata), the client signature, the proposal (input of the chaincode), the response (output of the chaincode), and the endorsement (list of transaction responses from other organizations).

There are four main reasons for choosing this blockchain framework. HLF "enables performance at scale while preserving privacy" [100] while still being customizable and modular. Secondly, it is one of the most used blockchain frameworks and is open-source, having multiple scientific studies, (as mentioned in subsections 3.3.3 and 4.3.1), 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 section 4.3.1. 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." [101]. In section 4.3.1, this technology usage is explained.

The distributed event streaming platform chosen for the data stream pipeline, that will manage the messages arrival for analysis, is Apache Kafka [102]. 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. However, if needed, more Zookeeper servers can be added, as long as in odd numbers, creating a Zookeeper Cluster. In production, three or five Zookeeper servers are advised but in the proof-of-concept only one is used.

River ML 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." [103].

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 were implemented using Python 3.8. All of the above mentioned technologies are free and open-source.

5.2.2 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, using a virtualization technology, virt-manager, twelve Virtual Machines were created with the following specifications.

CPU: 1 virtualCPU

RAM: 4GB

HDD: 30GB

OS Version: Ubuntu Desktop 20.04.4 LTS

5.2.3 Proof-of-Concept Software Architecture

An internal network, with Network Address Translation (NAT), was also implemented for communication between said machines. This network was named Olympus. In Figure 5.4 we can see the HLF implementation representation.

Alpha organization is composed of four Virtual Machine (VM)s, AlphaCA (the Certificate Authority), AlphaAdmin (the organization Admin), and the two organization 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 orderer peers - 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 peer and the orderer nodes for proper function. Figure 5.5 showcases IPFS representation. The VMs used for the HLF peers are also used as peers for the IPFS cluster, Zeus, Poseidon and Hera.

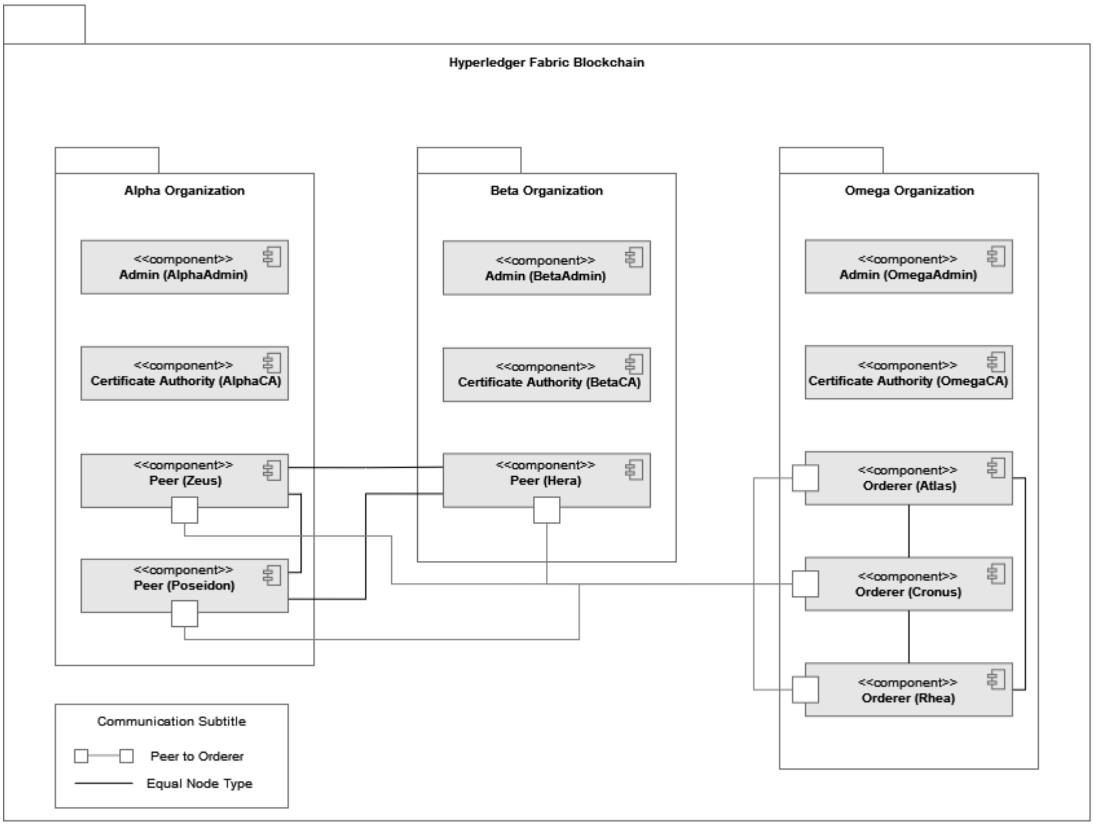

Figure 5.4: HLF Proof-of-Concept, UML Component Diagram

Figure 5.6 shows the data stream pipeline representation. 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 other. For an easier implementation of adding data, one of the brokers should be 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 5.7 shows the overral system at runtime.

5.2.4 Proof-of-Concept Implementation

In Annex 6.4, the proof-of-concept code is presented. The chaincode 1 implemented to allow CRUD (Create, Read, Update, Delete) actions in HLF blockchain is presented. Although not used in the proof-of-concept, it is possible to delete the contents of IPFS, cid, and keep the block 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

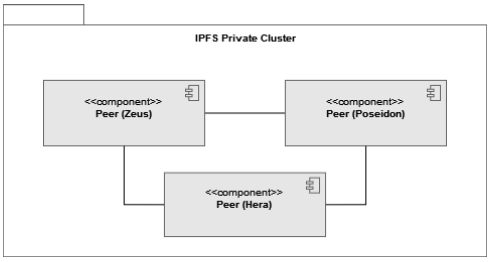

Figure 5.5: IPFS Proof-of-Concept, UML Component Diagram

Figure 5.6: Data Stream Pipeline Proof-of-Concept, UML Component Diagram

added data to the chosen Kafka topic. The HLF, the IPFS and the Kafka Producer are controlled with bash commands, through the python programs 2, 3, 4. All the bash commands are executed with the subprocess library 6. Hera's main controller is presented 5 and is responsible for integrating all these functionalities.

AlphaCA VM is responsible for receiving and analysing data sent to a given kafka topic 8. The analysis is made with the river python library 7. All the bash commands are executed with the subprocess library 6. AlphaCA's main controller is presented 9 and is responsible for integrating all these functionalities.

5.3 Evaluation

For the evaluation of the proof-of-concept, two data sources are used. The proof-of-concept design allows any type of data to be analyzed. As such, the proposed architecture is cross-application. The test environment is the same as the demonstration implementation environment 5.2.2. To measure the time different components of the system take, the timeit python library is used 11.

The first data source used were random files created with 10 bytes, 10 kilobytes, 1 megabyte using

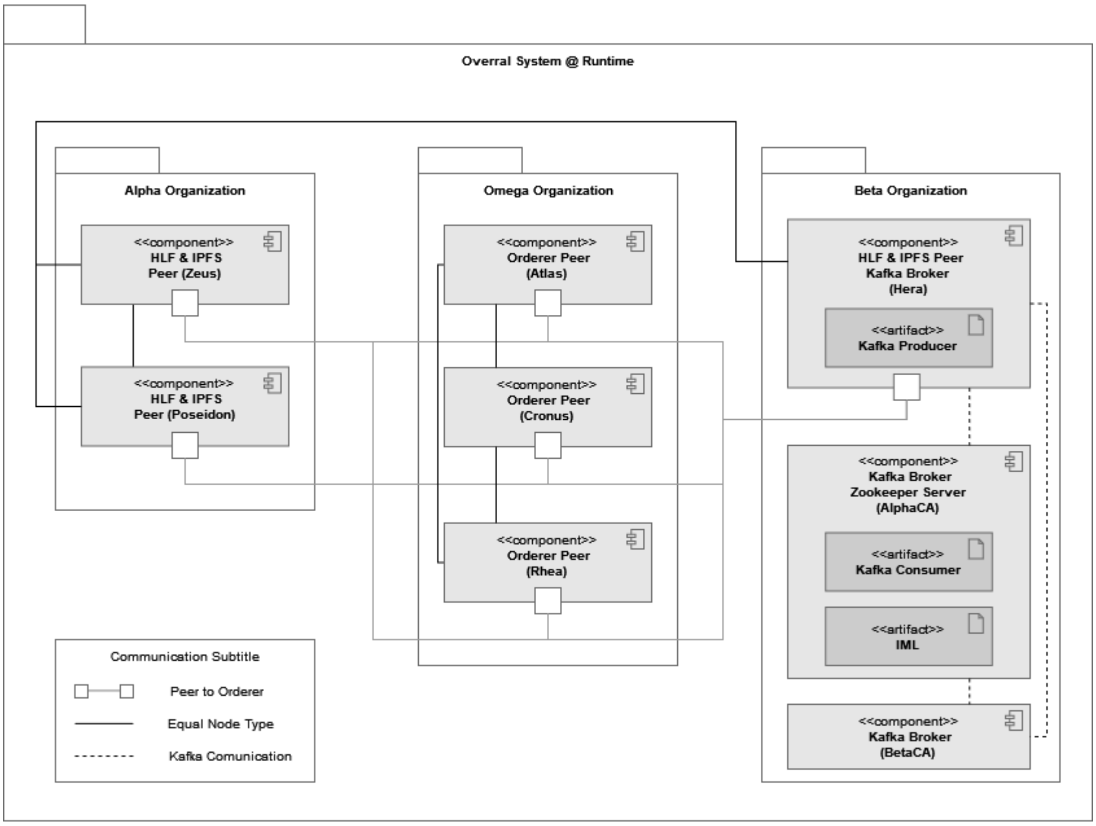

Figure 5.7: Overral System at Runtime, UML Component Diagram

the code in Listing 10. 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 a dataset about the insurance costs of healthcare in the USA and was extracted from Kaggle [104]. 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.

5.3.1 Produced Dataset

The Figures 5.8 and 5.9 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 5.10 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 scales linearly, in regard to the number of files.

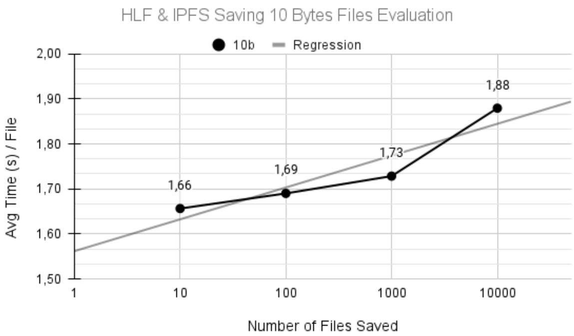

Figure 5.8: HLF & IPFS 10 Bytes Files Evaluation

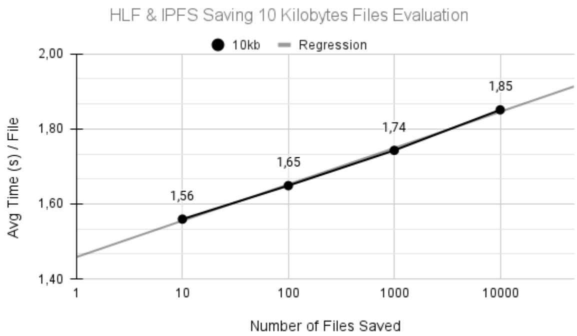

Figure 5.9: HLF & IPFS 10 Kilobytes Files Evaluation

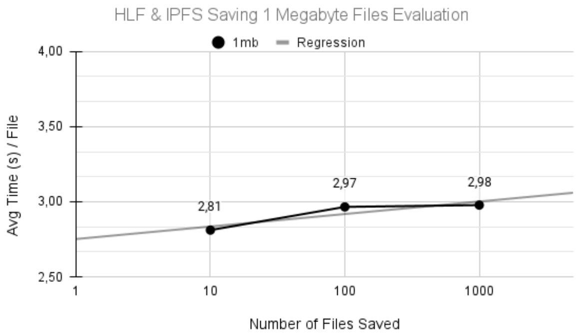

Figure 5.10: HLF & IPFS 1 Megabyte Files Evaluation

The Figures 5.11 and 5.12 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 5.13 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 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 regard to the number of files.

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.

The Figures 5.14 and 5.15 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 5.13 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 time measured in the Kafka consumer are impacted by the Kafka 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

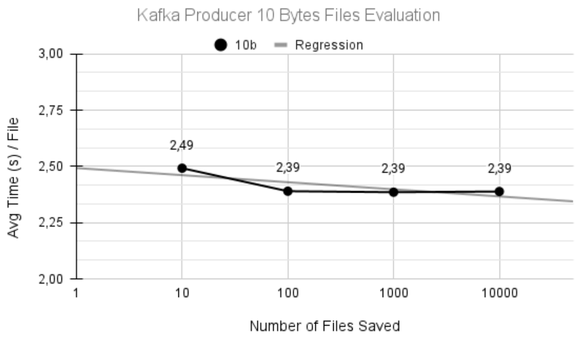

Figure 5.11: Kafka Cluster Producer 10 Bytes Files Evaluation

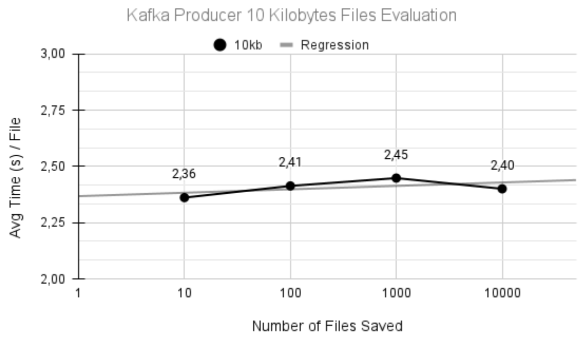

Figure 5.12: Kafka Cluster Producer 10 Kilobytes Files Evaluation

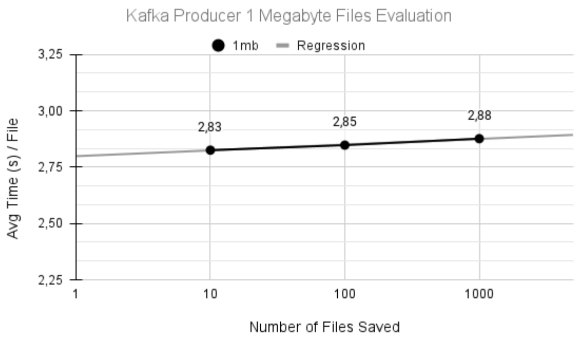

Figure 5.13: Kafka Cluster Producer 1 Megabyte Files Evaluation

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 has a low impact on 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, in regard to the number of files.

5.3.2 Healthcare Dataset

The healthcare insurance cost dataset [104] 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 ML metrics for linear regressions. The model is saved every 50 data entries.

Using the above mentioned dataset, two approaches were used. The first scenario tests the time it

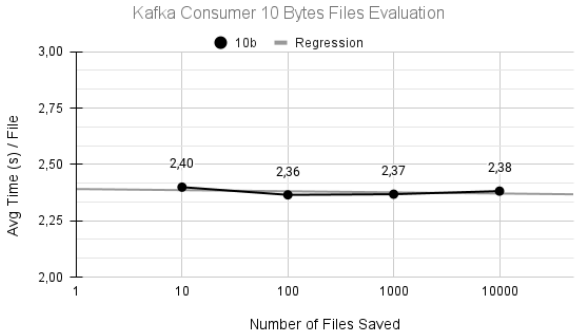

Figure 5.14: Kafka Cluster Consumer 10 Bytes Files Evaluation

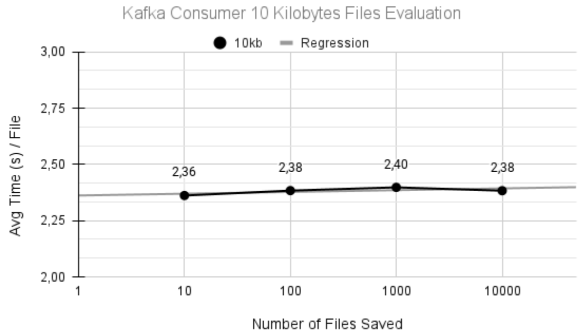

Figure 5.15: Kafka Cluster Consumer 10 Kilobytes Files Evaluation

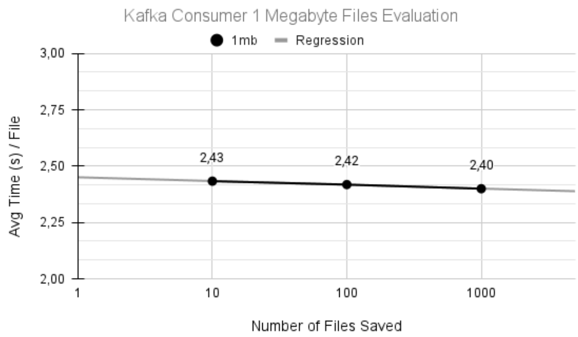

Figure 5.16: Kafka Cluster Consumer 1 Megabyte Files Evaluation

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 average time 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, 28 minutes 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, or better data mining. Since these scenarios goal is to measure the duration of the analysis, (and showcase the architecture capabilities), the analysis results (metrics) were not optimized.

5.4 Discussion

In this chapter, 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 pre-processing 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: demonstrate the architecture usage with existing technologies; showcase how it would scale; demonstrate, with a real-world dataset, the ability to make predictions from data while continuously improving a simple incremental machine learning model.

There are multiple possible applications for our proposed architecture. Using a similar implementation to the proof-of-concept, we present examples of applications in specific industries:

- Weather-dependent Industries (e.g., Agriculture, Fishing, Mining, Construction) – Weather data could be aggregated in the blockchain, with contributions from public participants, and each organization could extract value from the data by analysing it. After an initial setup, the model would continuously learn, improve and make predictions tailored to the organization's use case.

- Healthcare – Medical data could be aggregated with blockchain from different participants (e.g., hospitals, research centers, and clinics), and users could control how much information they allow each organization type to access for analysis, with minor changes to the smart contracts and the API. For example, research centers would have access to large amounts of private data for research purposes and could develop constantly improving models that hospitals could use to predict patients' health problems.
- Energy & Smart Cities – Energy consumers could provide data on consumption to the blockchain, and energy producers could leverage it to create models to predict needs and more accurately adjust energy production and price.
- Supply Chains – A customer-driven production and supply process, pull supply chains, involve integrating and using multiple platforms from different organizations. It can be difficult to exchange data between them and sometimes data can be lost in the process. However, using a similar implementation of our proof-of-concept, it is possible to aggregate the data and, with the proposed method, obtain predictions on consumer demand across the supply chain providing every organization with this critical information. In addition, the model would continuously improve and provide readily available updated predictions.

5.5 Related Work

To the best of our knowledge, not many studies propose architectures to analyse data stored in a blockchain. A simple approach to this problem would be analysing the data directly in the ledger using smart contracts and reading all the data every time an analysis is necessary. Since extracting potentially large amounts of data from the blockchain every time an analysis is required could be a time-consuming process, another approach entails saving data to a data warehouse and analysing it with batch-based machine learning. When data is added to the blockchain, the data warehouse would need to be updated. However, this approach trades increased performance for the risk of data integrity since the immutability of the blockchain no longer protects it. However, some cryptographic protections can be put in place to detect tampering. When compared with such implementations, the main advantages of our proposed architecture are:

- Maintains data integrity — Analysis are made on the data source, blockchain, and the DFS;

- Maintains the analysis model updated – Every time data is saved, it is sent to the analysis pipeline;
- Fast and up-to-date predictions – The most recent model is always available to make predictions;

Other studies that discuss the topic of data analysis in blockchain such as Bandara [37], present a solution that although efficient is limited to the blockchain the authors created not expanding to public blockchains. It is a solution that scales better than our proposal but does not establishes a framework for data analysis in blockchain and cannot be used for multiple applications. When compared to other implementations, our architecture is:

- Customizable – Modules are loosely coupled allowing for custom implementations where components may be removed according the application purpose;
- Flexible – It is possible to use different technologies, such as different types of blockchain, to better suit the application purposes;
- Adaptable – By expanding the API of a system, it is possible to implement our data analysis pipeline in blockchains already being used.
- Enables data aggregation – Depending on the implementation, data from multiple data sources and organizations can be collected, stored, and shared;
- Further, in use cases where multiple blockchain participants have access to the data, each can create information from the data source by making different analysis;

6

Conclusion

Contents

Given the recent increase in blockchain systems popularity, there are several proposals that explore this technology applications in multiple fields. Through this work, we have answered six research questions and organized the current body of knowledge identified. Additionally, the proposed architecture presents a potential solution for some of the identified issues, such as, the lack of data analysis solutions over blockchain, the time it takes to analyze said data and how new and old systems moving forward could collect, store and analyze said data securely. It also presents itself as a solution for a data collection system to feed a data analytics platform, through the blockchain.

6.1 Main Contributions

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. With this goal, we used a Systematic Literature Review to identify the main (1) benefits and (2) challenges, and possible (3) solutions for data analysis on the blockchain.

With the knowledge extracted from the above research, besides the research questions answers, insights such as the use of a distributed file system were learned. In DFS the amount, speed and type of data would be improved. Also, streaming data technologies allow for a higher data flow from the moment data is accessed to the analysis.

The second 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, we proposed an architecture 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, we included a data stream pipeline 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).

We developed a software proof-of-concept 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 River ML is utilized as a incremental machine learning tool.

We evaluated the proof-of-concept using two datasets: 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.

6.2 Communication

This thesis manuscript and the scientific manuscripts produced, "Benefits, Challenges, and Solutions for Data Analysis in Blockchain: Review" and "Data Analysis in Blockchain: Review & Architecture" submitted to academic journals for publication represent the communication phase of the DSRM. Through the SLRs, 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.

6.3 Research Limitations

The review's 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.

6.4 Future Work

Future work could implement and test prototypes with different technologies (e.g., blockchain frameworks, distributed file systems, incremental machine learning libraries, and data streaming platforms). It would be interesting to better address performance with load testing, spike testing, stress testing, volume testing, endurance testing, and scalability testing. For a specific use case, it would be interesting to compare the proposed solution with a classical one based on a data warehouse and batch-based machine learning data analysis.

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] M. S. Brown, “Transforming unstructured data into useful information,” *Big Data, Mining, and Analytics: Components of Strategic Decision Making*, 2014.

[5] J. A. Jaoude and R. G. Saade, “Blockchain applications - usage in different domains,” *IEEE Access*, vol. 7, pp. 45 360–45 381, 2019.

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

[7] A. R. Hevner, S. T. March, J. Park, and S. Ram, “Design sceince in information systems,” *MIS Quarterly*, vol. 28, pp. 75–105, 2004.

[8] 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.

[9] H. Y. Paik, X. Xu, H. M. Bandara, S. U. Lee, and S. K. Lo, “Analysis of data management in blockchain-based systems: From architecture to governance,” *IEEE Access*, vol. 7, pp. 186 091–186 107, 2019.

[10] 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.

[11] B. E. Blakely, P. Pawar, L. Jololian, and S. Prabhaker, “The convergence of EDI, blockchain, and big data in health care,” *Conference Proceedings - IEEE SOUTHEASTCON*, vol. 2021-March, 2021.

[12] C. Edussuriya, K. Vithanage, N. Bandara, J. Alawatugoda, M. Sandirigama, U. Jayasinghe, N. Shone, and G. M. Lee, “BAT—block analytics tool integrated with blockchain based iot platform,” *Electronics (Switzerland)*, vol. 9, no. 9, pp. 1–20, 2020.

[13] F. Jamil, F. Qayyum, S. Alhelaly, F. Javed, and A. Muthanna, “Intelligent microservice based on blockchain for healthcare applications,” *Computers, Materials and Continua*, vol. 69, no. 2, pp. 2513–2530, 2021.

[14] A. Kumari and S. Tanwar, “A Data Analytics Scheme for Security-aware Demand Response Management in Smart Grid System,” *7th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering, UPCON 2020*, 2020.

[15] G. J. Mendis, Y. Wu, J. Wei, M. Sabounchi, and R. Roche, “A Blockchain-Powered Decentralized and Secure Computing Paradigm,” *IEEE Transactions on Emerging Topics in Computing*, vol. 6750, no. c, pp. 1–18, 2020.

[16] N. V. Pardakhe and V. M. Deshmukh, “Machine Learning and Blockchain Techniques Used in Healthcare System,” *2019 IEEE Pune Section International Conference, PuneCon 2019*, pp. 1–5, 2019.

[17] M. A. Rahman, M. Rashid, S. Barnes, M. Shamim Hossain, E. Hassanain, and M. Guizani, “An IoT and blockchain-based multi-sensory in-home quality of life framework for cancer patients,” *2019 15th International Wireless Communications and Mobile Computing Conference, IWCMC 2019*, pp. 2116–2121, 2019.

[18] M. A. Rahman, M. M. Rashid, J. L. Kernec, B. Philippe, S. J. Barnes, F. Fioranelli, S. Yang, O. Romain, Q. H. Abbasi, G. Loukas, and M. Imran, “A secure occupational therapy framework for monitoring cancer patients’ quality of life,” *Sensors (Switzerland)*, vol. 19, no. 23, 2019.

[19] A. Raslan, G. Kapogiannis, A. Cheshmehzangi, W. Tizani, and D. Towey, “A Framework for Assembling Asset Information Models (AIMs) through Permissioned Blockchain,” *Proceedings - 2020 IEEE 44th Annual Computers, Software, and Applications Conference, COMPSAC 2020*, pp. 529–534, 2020.

[20] S. Rasool, M. Iqbal, T. Dagiuklas, Z. Ul-Qayyum, and S. Li, “Reliable Data Analysis through Blockchain based Crowdsourcing in Mobile Ad-hoc Cloud,” *Mobile Networks and Applications*, vol. 25, no. 1, pp. 153–163, 2020.

[21] C. Schaefer and C. Edman, "Transparent logging with hyperledger fabric," *ICBC 2019 - IEEE International Conference on Blockchain and Cryptocurrency*, pp. 65–69, 2019.

[22] Z. Shae and J. J. Tsai, "On the Design of a Blockchain Platform for Clinical Trial and Precision Medicine," *Proceedings - International Conference on Distributed Computing Systems*, pp. 1972–1980, 2017.

[23] N. Sukhija, E. Bautista, M. Moore, and J. G. Sample, "Employing blockchain technology for decentralized crowdsourced data access and management," *Proceedings - 2019 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Internet of People and Smart City Innovation, SmartWorld/UIC/ATC/SCALCOM/IOP/SCI 2019*, pp. 268–273, 2019.

[24] S. Tanwar, Q. Bhatia, P. Patel, A. Kumari, P. K. Singh, and W. C. Hong, "Machine Learning Adoption in Blockchain-Based Smart Applications: The Challenges, and a Way Forward," *IEEE Access*, vol. 8, pp. 474–448, 2020.

[25] L. Zhu and F. Li, "Agricultural data sharing and sustainable development of ecosystem based on block chain," *Journal of Cleaner Production*, vol. 315, no. April, p. 127869, 2021. [Online]. Available: https://doi.org/10.1016/j.jclepro.2021.127869

[26] D. Dhagarra, M. Goswami, P. R. Sarma, and A. Choudhury, "Big Data and blockchain supported conceptual model for enhanced healthcare coverage: The Indian context," *Business Process Management Journal*, vol. 25, no. 7, pp. 1612–1632, 2019.

[27] K. Lampropoulos, G. Georgakakos, and S. Ioannidis, "Using blockchains to enable big data analysis of private information," *IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD*, vol. 2019-Septe, 2019.

[28] Z. Shahbazi and Y. C. Byun, "Improving transactional data system based on an edge computing-blockchain-machine learning integrated framework," *Processes*, vol. 9, no. 1, pp. 1–20, 2021.

[29] M. Nugent and R. G. Lennon, "Blockchain for Decentralized Data Analysis," *INFOCOM 2019 - IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2019*, pp. 1059–1060, 2019.

[30] S. Venkatraman and R. Venkatraman, "Big data security challenges and strategies," *AIMS Mathematics*, vol. 4, no. 3, pp. 860–879, 2019.

[31] A. Choubey, S. Behera, Y. S. Patel, K. Mahidhar, and R. Misra, "EnergyTradingRank Algorithm for Truthful Auctions among EVs via Blockchain Analytics of Large Scale Transaction Graphs," *2019*

11th International Conference on Communication Systems and Networks, COMSNETS 2019, pp. 9–14, 2019.

[32] J. Xiaomeng, Z. Fan, L. Shenwen, Y. Jinglin, and H. Ketai, "Data Analysis of Bitcoin Blockchain Network Nodes," *Proceedings of the 15th IEEE Conference on Industrial Electronics and Applications, ICIEA 2020*, pp. 1891–1895, 2020.

[33] P. Zheng, Z. Zheng, J. Wu, and H.-N. Dai, "XBlock-ETH: Extracting and Exploring Blockchain Data From Ethereum," *IEEE Open Journal of the Computer Society*, vol. 1, no. April, pp. 95–106, 2020.

[34] V. G. Venkatesh, K. Kang, B. Wang, R. Y. Zhong, and A. Zhang, "System architecture for blockchain based transparency of supply chain social sustainability," *Robotics and Computer-Integrated Manufacturing*, vol. 63, no. November 2019, p. 101896, 2020. [Online]. Available: https://doi.org/10.1016/j.rcim.2019.101896

[35] L. Li and H. Tong, "Blockchain Data Analytics," *Comp.Hkbu.Edu.Hk*, 2018.

[36] "European commission," last accessed 30 December 2021. [Online]. Available: https://ec.europa.eu/info/law/law-topic/data-protection/

[37] E. Bandara, X. Liang, P. Foytik, S. Shetty, N. Ranasinghe, and K. De Zoysa, "Rahasak—Scalable blockchain architecture for enterprise applications," *Journal of Systems Architecture*, vol. 116, no. February, p. 102061, 2021. [Online]. Available: https://doi.org/10.1016/j.sysarc.2021.102061

[38] A. Learning, "Storing and Querying Bitcoin Blockchain Using SQL Databases," *Information Systems Education Journal (ISEDJ)*, vol. 12, no. 4, p. 6, 2014.

[39] Y. Gao, H. Lin, Y. Chen, and Y. Liu, "Blockchain and SGX-Enabled Edge-Computing-Empowered Secure IoMT Data Analysis," *IEEE Internet of Things Journal*, vol. 8, no. 21, pp. 15 785–15 795, 2021.

[40] M. Moniruzzaman, S. Khezr, A. Yassine, and R. Benlamri, "Blockchain for smart homes: Review of current trends and research challenges," *Computers and Electrical Engineering*, vol. 83, no. 2020, 2020.

[41] H. Bi, J. Liu, and N. Kato, "Deep Learning-based Privacy Preservation and Data Analytics for IoT Enabled Healthcare," *IEEE Transactions on Industrial Informatics*, vol. 3203, no. c, pp. 1–10, 2021.

[42] D. Unal, M. Hammoudeh, M. A. Khan, A. Abuarqoub, G. Epiphaniou, and R. Hamila, "Integration of federated machine learning and blockchain for the provision of secure big data analytics for Internet of Things," *Computers and Security*, vol. 109, p. 102393, 2021. [Online]. Available: https://doi.org/10.1016/j.cose.2021.102393

[43] Y. Wu, P. Zheng, J. Guo, W. Zhang, and J. Huang, “A Controllable Efficient Content Distribution Framework Based on Blockchain and ISODATA,” *Proceedings - 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications and 12th IEEE International Conference on Big Data Science and Engineering, Trustcom/BigDataSE 2018*, pp. 1698–1701, 2018.

[44] Y. Wu, Z. Wang, Y. Ma, and V. C. Leung, “Deep reinforcement learning for blockchain in industrial IoT: A survey,” *Computer Networks*, vol. 191, no. February, 2021.

[45] E. Zhou, H. Sun, B. Pi, J. Sun, K. Yamashita, and Y. Nomura, “Ledgerdata Refiner: A Powerful Ledger Data Query Platform for Hyperledger Fabric,” *2019 6th International Conference on Internet of Things: Systems, Management and Security, IOTSMS 2019*, pp. 433–440, 2019.

[46] W. T. Tsai, R. Wang, S. Liu, E. Deng, and D. Yang, “COMPASS: A data-driven blockchain evaluation framwework,” *Proceedings - 14th IEEE International Conference on Service-Oriented System Engineering, SOSE 2020*, pp. 17–30, 2020.

[47] D. T. Jose and A. B. Technology, “TOTEM : Token for controlled computation,” *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–7, 2019.

[48] R. Galici, L. Ordile, M. Marchesi, A. Pinna, and R. Tonelli, “Applying the ETL process to blockchain data. Prospect and findings,” *Information (Switzerland)*, vol. 11, no. 4, pp. 1–15, 2020.

[49] N. Wan, Y. Liu, and W. Xiao, “A Financial Transaction Methods Based on MapReduce Technology and Blockchain,” *Proceedings - 2020 3rd International Conference on Smart BlockChain, SmartBlock 2020*, pp. 109–113, 2020.

[50] Q. Zheng, Y. Li, P. Chen, and X. Dong, “An innovative ipfs-based storage model for blockchain,” *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pp. 704–708, 2018.

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

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

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

[54] 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.

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

[56] 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.

[57] 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].

[58] 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.

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

[60] 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.

[61] 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.

[62] 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.

[63] 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.

[64] 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.

[65] 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.

[66] 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.

[67] 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.

[68] 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.

[69] 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.

[70] 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.

[71] 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.

[72] 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.

[73] 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.

[74] 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.

[75] 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.

[76] 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.

[77] 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.

[78] 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.

[79] 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.

[80] 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.

[81] 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.

[82] A. Tandon, A. Dhir, A. N. Islam, and M. Mäntymäki, “Blockchain in healthcare: A systematic literature review, synthesizing framework and future research agenda,” *Computers in Industry*, vol. 122, p. 103290, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166361520305248

[83] Y. Wang, J. H. Han, and P. Beynon-Davies, “Understanding blockchain technology for future supply chains: a systematic literature review and research agenda,” *Supply Chain Management: An International Journal*, 2018.

[84] H. Khan and T. Masood, “Impact of blockchain technology on smart grids-a systematic literature review,” *Available at SSRN 4003063*, 2021.

[85] S. K. Lo, Y. Liu, S. Y. Chia, X. Xu, Q. Lu, L. Zhu, and H. Ning, “Analysis of blockchain solutions for iot: A systematic literature review,” *IEEE Access*, vol. 7, pp. 58 822–58 835, 2019.

[86] C. Shen and F. Pena-Mora, “Blockchain for cities—a systematic literature review,” *Ieee Access*, vol. 6, pp. 76 787–76 819, 2018.

[87] O. Ali, M. Ally, Y. Dwivedi *et al.*, “The state of play of blockchain technology in the financial services sector: A systematic literature review,” *International Journal of Information Management*, vol. 54, p. 102199, 2020.

[88] F. R. Batubara, J. Ubacht, and M. Janssen, “Challenges of blockchain technology adoption for e-government: a systematic literature review,” *Proceedings of the 19th Annual International Conference on Digital Government Research: Governance in the Data Age*, pp. 1–9, 2018.

[89] F. Loukil, M. Abed, and K. Boukadi, “Blockchain adoption in education: a systematic literature review,” *Education and Information Technologies*, vol. 26, no. 5, pp. 5779–5797, 2021.

[90] O. Bermeo-Almeida, M. Cardenas-Rodriguez, T. Samaniego-Cobo, E. Ferruzola-Gomez, R. Cabezas-Cabezas, and W. Bazan-Vera, "Blockchain in agriculture: A systematic literature review," *International Conference on Technologies and Innovation*, pp. 44–56, 2018.

[91] P. J. Taylor, T. Dargahi, A. Dehghantanha, R. M. Parizi, and K. K. R. Choo, "A systematic literature review of blockchain cyber security," *Digital Communications and Networks*, vol. 6, pp. 147–156, 5 2020.

[92] F. J. de Haro-Olmo, Ángel Jesús Varela-Vaca, and J. A. Álvarez Bermejo, "Blockchain from the perspective of privacy and anonymisation: A systematic literature review," *Sensors (Switzerland)*, vol. 20, pp. 1–21, 12 2020.

[93] H. Huang, J. Lin, B. Zheng, Z. Zheng, and J. Bian, "When blockchain meets distributed file systems: An overview, challenges, and open issues," *IEEE Access*, vol. 8, pp. 50 574–50 586, 2020.

[94] N. Deepa, Q.-V. Pham, D. C. Nguyen, S. Bhattacharya, B. Prabadevi, T. R. Gadekallu, P. K. R. Maddikunta, F. Fang, and P. N. Pathirana, "A survey on blockchain for big data: approaches, opportunities, and future directions," *Future Generation Computer Systems*, 2022.

[95] A. A. Hussain and F. Al-Turjman, "Artificial intelligence and blockchain: A review," *Transactions on Emerging Telecommunications Technologies*, vol. 32, 9 2021.

[96] Y. Luo, L. Yin, W. Bai, and K. Mao, "An appraisal of incremental learning methods," *Entropy*, vol. 22, no. 11, p. 1190, 2020.

[97] 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.

[98] "Linux foundation team," last accessed 16 October 2022. [Online]. Available: https://www.linuxfoundation.org/

[99] "Deploy the ordering service," last accessed 16 October 2022. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.4/deployorderer/ordererdeploy.html#deploy-the-ordering-service

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

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

[102] "Apache kafka team," last accessed 16 October 2022. [Online]. Available: https://kafka.apache.org/

[103] “River ml team,” last accessed 16 October 2022. [Online]. Available: https://riverml.xyz/

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

Appendix A: Proof of Concept Code

Listing 1: `chaincode.go`

```
package main

import {
	"enconding/json"
	"fmt"
	"log"
	"time"
	"strings"

	"github.com/hyperledger/fabric-contract-api-go/contractapi"
}

// SmartContract provides functions for managing an Asset
type SmartContract struct {
	contractapi.Contract
}

// Asset describes basic details of what makes up a simple asset
type Asset struct {
	ID              string  `json:"ID"`
	TimeStamp       string  `json:"timeStamp"`
	CID             string  `json:"CID"`
	Hash            string  `json:"hash"`
	Deleted         bool    `json:"deleted"`
}

```

```
// CreateAsset issues a new asset to the world state with given details.
func (s *SmartContract) CreateAsset(ctx contractapi.TransactionContextInterface, id string, cid string, hash string) error {
	exists, err := s.AssetExists(ctx, id)
	if err != nil {
		return err
	}
	if exists {
		return fmt.Errorf("the asset %s already exists", id)
	}
	t := time.Now()
	timestamp := t.Format("2006-01-02 15:04:05")

	asset := Asset{
		ID:             id,
		TimeStamp:      timestamp,
		CID:            cid,
		Hash:           hash,
		Deleted:        false,
	}
	assetJSON, err := json.Marshal(asset)
	if err != nil {
		return err
	}

	return ctx.GetStub().PutState(id, assetJSON)
}

// ReadAsset returns the asset stored in the world state with given id.
func (s *SmartContract) ReadAsset(ctx contractapi.TransactionContextInterface, id string) (*Asset, error) {
	assetJSON, err := ctx.GetStub().GetState(id)
	if err != nil {
		return nil, fmt.Errorf("failed to read from world state: %v", err)
	}
	if assetJSON == nil {
		return nil, fmt.Errorf("the asset %s does not exist", id)
	}
```

```go

    var asset Asset
    err = json.Unmarshal(assetJSON, &asset)
    if err != nil {
      return nil, err
    }

    return &asset, nil
}

// UpdateAsset updates an existing asset in the world state with provided
    parameters.
func (s *SmartContract) UpdateAsset(ctx contractapi.
    TransactionContextInterface, id string, cid string, hash string) error {
    exists, err := s.AssetExists(ctx, id)
    if err != nil {
      return err
    }
    if !exists {
      return fmt.Errorf("the asset %s does not exist", id)
    }
    t := time.Now()
    timestamp := t.Format("2006-01-02 15:04:05")

    // overwriting original asset with new asset
    asset := Asset{
      ID:             id,
      TimeStamp:      timestamp,
      CID:            cid,
      Hash:           hash,
      Deleted:        false,
    }

    assetJSON, err := json.Marshal(asset)
    if err != nil {
      return err
    }

```

```
    return ctx.GetStub().PutState(id, assetJSON)
}

// DeleteAsset deletes an given asset from the world state.
func (s *SmartContract) DeleteAsset(ctx contractapi.
    TransactionContextInterface, id string) error {
    exists, err := s.AssetExists(ctx, id)
    if err != nil {
      return err
    }
    if !exists {
      return fmt.Errorf("the asset %s does not exist", id)
    }

    return ctx.GetStub().DelState(id)
}

// AssetExists returns true when asset with given ID exists in world state
func (s *SmartContract) AssetExists(ctx contractapi.
    TransactionContextInterface, id string) (bool, error) {
    assetJSON, err := ctx.GetStub().GetState(id)
    if err != nil {
      return false, fmt.Errorf("failed to read from world state: %v", err)
    }

    return assetJSON != nil, nil
}

//When the correspondent ipfs file is deleted, update the deleted parameter.
func (s *SmartContract) SetDeleted(ctx contractapi.
    TransactionContextInterface, id string) error {
    asset, err := s.ReadAsset(ctx, id)
    if err != nil {
      return err
    }

    asset.Deleted = true
    assetJSON, err := json.Marshal(asset)
```

```
    if err != nil {
      return err
    }

    return ctx.GetStub().PutState(id, assetJSON)
}

// GetAllAssets returns all assets found in world state
func (s *SmartContract) GetAllAssets(ctx contractapi.TransactionContextInterface) ([]*Asset, error) {
// range query with empty string for startKey and endKey does an
// open-ended query of all assets in the chaincode namespace.
    resultsIterator, err := ctx.GetStub().GetStateByRange("", "")
    if err != nil {
      return nil, err
    }
    defer resultsIterator.Close()

    var assets []*Asset
    for resultsIterator.HasNext() {
      queryResponse, err := resultsIterator.Next()
      if err != nil {
        return nil, err
      }

      var asset Asset
      err = json.Unmarshal(queryResponse.Value, &asset)
      if err != nil {
        return nil, err
      }
      assets = append(assets, &asset)
    }

    return assets, nil
}

func main() {
    assetChaincode, err := contractapi.NewChaincode(&SmartContract{})
```

```
    if err != nil {
      log.Panicf("Error creating asset-transfer-basic chaincode: %v", err)
    }

    if err := assetChaincode.Start(); err != nil {
      log.Panicf("Error starting asset-transfer-basic chaincode: %v", err)
    }
  }
```

Listing 2: `hlf.py`

```
import ipfs
from executeCmd import execute_cmd

IPFS_FILES_PATH = "~/IPFS/files/"
HLF_PATH = "~/HLF/fabric/bin/"

def add_data_hlf(id, filename):
    """
    receives the unique ID and the filename with path e.g: ~/add.txt
    return
    """
    cid = ipfs.add_data_ipfs(filename)
    cmd = HLF_PATH + "peer chaincode invoke -o atlas.omega.olympus.pt:7050 --
        tls --cafile ~/HLF/organizations/ordererOrganizations/omega.olympus.
        pt/msp/tlscacerts/ca-omega-olympus-pt-7054.pem -C main-channel -n
        occv3 -c '{\"Args\":[\"CreateAsset\",\"" + str(id) + "\",\"" + str(
        cid) + "\"]}'"
    response = execute_cmd(cmd)
    if "status:200" in response:
        return True
    else:
        return False

def delete_data_hlf(id):
    """
    receives the id
    """
```

```
    cmd = HLF_PATH + "peer chaincode invoke -o atlas.omega.olympus.pt:7050 --tls --cafile ~/HLF/organizations/ordererOrganizations/omega.olympus.pt/msp/tlscacerts/ca-omega-olympus-pt-7054.pem -C main-channel -n occv3 -c '{\"Args\":[\"ReadAsset\",\"" + str(id) + "\"]}'"
    response = execute_cmd(cmd)
    if "status:200" in response:
        lst_response = response.split("CID")
        response_cid = lst_response[-1]
        lst_response = response_cid.split("deleted")
        response_cid = lst_response[0]
        ipfs_cid = ""
        for char in response_cid:
            if char != "/" and char != "\"" and char != "\\" and char != ":" and char != ",":
                ipfs_cid = ipfs_cid + char

        if (ipfs.delete_data_ipfs(ipfs_cid) == True):
            pass
        else:
            return False
    else:
        return False

    cmd = HLF_PATH + "./peer chaincode invoke -o atlas.omega.olympus.pt:7050 --tls --cafile ~/HLF/organizations/ordererOrganizations/omega.olympus.pt/msp/tlscacerts/ca-omega-olympus-pt-7054.pem -C main-channel -n occv3 -c '{\"Args\":[\"DeleteAsset\",\"" + str(id) + "\"]}'"
    response = execute_cmd(cmd)
    if "status:200" in response:
        return True
    else:
        return False

def get_data_hlf(id):
    """
    receives the blockid
    return data_path stored in IPFS
    """
```

```python
    cmd = HLF_PATH + "peer chaincode invoke -o atlas.omega.olympus.pt:7050 --tls --cafile ~/HLF/organizations/ordererOrganizations/omega.olympus.pt/msp/tlscacerts/ca-omega-olympus-pt-7054.pem -C main-channel -n occv3 -c '{\"Args\":[\"ReadAsset\",\"" + str(id) + "\"]}'"
    response = execute_cmd(cmd)
    if "status:200" in response:
        lst_response = response.split("CID")
        response_cid = lst_response[-1]
        lst_response = response_cid.split("deleted")
        response_cid = lst_response[0]
        ipfs_cid = ""
        for char in response_cid:
            if char != "/" and char != "\"" and char != "\\" and char != ":" and char != ",":
                ipfs_cid = ipfs_cid + char

        data_path = ipfs.get_data_ipfs(ipfs_cid)
    return data_path

def get_all_data_hlf():
    """
    returns path with all the data
    """
    cmd = HLF_PATH + "peer chaincode invoke -o atlas.omega.olympus.pt:7050 --tls --cafile ~/HLF/organizations/ordererOrganizations/omega.olympus.pt/msp/tlscacerts/ca-omega-olympus-pt-7054.pem -C main-channel -n occv3 -c '{\"Args\":[\"GetAllAssets\"]}'"
    response = execute_cmd(cmd)
    if "status:200" in response and "payload" in response:
        lst_response = response.split("CID")
        lst_response = lst_response[1:]
        ipfs_cid = ""
        for response_cid in lst_response:
            lst_response = response_cid.split("deleted")
            response_cid = lst_response[0]
            for char in response_cid:
                if char != "/" and char != "\"" and char != "\\" and char != ":" and char != ",":
```

```
                        ipfs_cid = ipfs_cid + char
                ipfs.get_data_ipfs(ipfs_cid)
                ipfs_cid = ""
    return IPFS_FILES_PATH
```

Listing 3: `ipfs.py`

```
from executeCmd import execute_cmd

IPFS_FILES_PATH = "~/IPFS/files/"

def add_data_ipfs(filename):
    """
    receives the filename in e.g: ~/mypath/add.txt
    return new content identifier
    """

    #convert input to string
    filename = str(filename)

    cmd = "cp " + filename + " " + IPFS_FILES_PATH
    execute_cmd(cmd)

    #add filename to ipfs and send hash to temporary file
    cmd = "ipfs-cluster-ctl add " + filename
    cmd_return = execute_cmd(cmd)

    #ipfs-cluster-ctl add return format -> added hash1010101 txt.txt
    list_cmd_return = cmd_return.split()
    ipfs_hash = list_cmd_return[1]

    cmd = "ipfs get " + ipfs_hash
    execute_cmd(cmd)

    cmd = "mv " + ipfs_hash + " " + IPFS_FILES_PATH
    execute_cmd(cmd)

    filename = filename.split("/")
```

```python
    cmd = "rm " + IPFS_FILES_PATH + filename[-1]
    execute_cmd(cmd)

    return ipfs_hash

def delete_data_ipfs(cid):
    """
    receives the content identifier of the data to be deleted
    """

    #convert input to string
    cid = str(cid)

    #unpin filename from ipfs and run garbage collector to delete from
        cluster
    cmd = "ipfs-cluster-ctl pin rm " + cid
    execute_cmd(cmd)

    cmd = "ipfs-cluster-ctl ipfs gc"
    execute_cmd(cmd)

    cmd = "rm " + IPFS_FILES_PATH + cid
    execute_cmd(cmd)

    return True

def get_data_ipfs(cid):
    """
    send ipfs content identifier
    return content path
    """

    #convert input to string
    cid = str(cid)

    #get data from ipfs
    cmd = "ipfs get " + cid
    execute_cmd(cmd)
```

```

    cmd = "mv " + cid + " " + IPFS_FILES_PATH
    execute_cmd(cmd)

    return IPFS_FILES_PATH + cid
```

Listing 4: `kafkaProducer.py`

```
from executeCmd import execute_cmd

KAFKA_PATH = "~/KAFKA/bin/"

def send_data(server_addr, topic_name, datafile):
    """
    send data to kafka topic
    server_addr can be 1 or more ip addrs serverA or serverA serverB
    data will be read each line at time
    """
    cmd = KAFKA_PATH + "kafka-console-producer.sh --broker-list " + str(
        server_addr) + ":9092 --topic " + str(topic_name) + " < " + str(
        datafile)
    execute_cmd(cmd)
    return True
```

Listing 5: `producerController.py`

```
import hlf
import kafkaProducer
from timeProgram import delay
import datasetController as dtsc
import signal

KAFKA_TOPIC = "trainer"
KAFKA_PREDICTION_TOPIC = "analyzer"
KAFKA_ADDR = "ca.alpha.olympus.pt"

CONTROLLER_FOLDER_PATH = "/home/hera/CONTROLLER/"
```

```python
DATA_FOLDER_PATH = CONTROLLER_FOLDER_PATH + "dataset_files/"
DATA_FOLDER_INDEX_PATH = DATA_FOLDER_PATH + "index.txt"
DATA_PREDICTION_FOLDER_PATH = CONTROLLER_FOLDER_PATH + "dataset_files_analysis/"
DATA_PREDICTION_FOLDER_INDEX_PATH = DATA_PREDICTION_FOLDER_PATH + "index.txt"

def handler(signum, frame):
    global flag
    res = input("ACTION: Ctrl-c was pressed. Do you really want to exit? y/n ")
    if res == 'y':
        flag = False

def add_datafiles():
    global flag
    flag = True
    signal.signal(signal.SIGINT, handler)
    hlf_id_tracker = 1

    dtsc.undo_selection()
    delay("Undo Selection")
    dtsc.random_selection_analysis(5)
    delay("Selection")

    with open(DATA_FOLDER_INDEX_PATH, "r") as indexFile:
        for line in indexFile:
            if(not flag):
                print("\nEXCEPTION: Stopped!\n")
                break
            filename = DATA_FOLDER_PATH + str(line)
            filename = filename[:-1] #removes last \n char
            add_data(hlf_id_tracker, filename)
            hlf_id_tracker += 1
            print("INFO: Added " + str(hlf_id_tracker) + " files\n")

    print("INFO: Last FileID Added is " + str(hlf_id_tracker))
    return hlf_id_tracker

```

```python
def add_datafiles_prediction():
    with open(DATA_PREDICTION_FOLDER_INDEX_PATH, "r") as indexFile:
        for line in indexFile:
            filename = DATA_PREDICTION_FOLDER_PATH + str(line)
            filename = filename[:-1] #removes last \n char
            send_prediction_data(filename)
            print("")
    return

def add_data(id, filename):
    status = hlf.add_data_hlf(id, filename)
    if not status:
        print('ERR: HLF or IPFS Error')
        return
    print('INFO: File Added With Success to HLF and IPFS')

    print('INFO: Sending Data to Kafka Cluster')
    response = kafkaProducer.send_data(KAFKA_ADDR, KAFKA_TOPIC, filename)
    if (response):
        print('INFO: Data Sent to Kafka Cluster')
    else:
        print('ERR: Data Not Sent to Kafka Cluster')
    return

def send_prediction_data(filename):
    if (kafkaProducer.send_data(KAFKA_ADDR, KAFKA_PREDICTION_TOPIC, filename)
        ):
        print('INFO: Data Sent to Kafka Cluster')
    else:
        print('ERR: Data Not Sent to Kafka Cluster')
    return

def read_data(id):
    data_path = hlf.get_data_hlf(id)
    print('INFO: Data Retrieved With Success from HLF and IPFS')
    print('INFO: Data Located in File: ' + data_path)
    return data_path

```

```
def read_all_data():
    data_path = hlf.get_all_data_hlf()
    print('INFO: All Data Fetched With Success from HLF and IPFS')
    print('INFO: All Data Located in Folder: ' + data_path)
    return data_path

def delete_data(id):
    status = hlf.delete_data_hlf(id)
    if not status:
        print('ERR: HLF or IPFS Error')
        return
    print('INFO: Data Deleted With Success from HLF and IPFS')
    return

def delete_all_data(id):
    id_counter = 1
    id = int(id)
    while (id_counter <= id):
        delete_data(id_counter)
        id_counter += 1
    print("INFO: All Data Deleted from HLF and IPFS")
```

Listing 6: `executeCmd.py`

```
import subprocess

def execute_cmd(cmd):
    ps = subprocess.Popen(cmd, shell=True, stdout=subprocess.PIPE, stderr=
        subprocess.STDOUT)
    response = str(ps.communicate()[0])
    return response
```

Listing 7: `iml.py`

```
from river import linear_model
from river import metrics
from river import compose
```

```
from river import preprocessing
from river import optim
import os
import pickle

SAVE_MODEL_FREQUENCY = 50

class Iml:

    def __init__(self):
        self.model = compose.Pipeline(
            ('cat_scale', compose.SelectType(str) | preprocessing.OneHotEncoder()),
            ('scale', compose.SelectType(int,float) |  preprocessing.StandardScaler()),
            ('log_reg', linear_model.LinearRegression(optimizer=optim.SGD(0.05)))
        )
        self.modelName = 'model.plk'
        self.metricMAE = metrics.MAE()
        self.metricRMSE = metrics.RMSE()
        self.metricR2 = metrics.R2()
        # save every counter interations
        self.counter = SAVE_MODEL_FREQUENCY

    def learn(self, entryData, targetData):
        x, y = entryData, targetData
        self.model.learn_one(x, y)
        y_pred = self.model.predict_one(x)

        self.metricMAE.update(y, y_pred)
        self.metricRMSE.update(y, y_pred)
        self.metricR2.update(y, y_pred)

        self.counter -= 1
        if (self.counter <= 0):
            print("\nINFO: Saving Model...\n")
            self.counter = SAVE_MODEL_FREQUENCY
```

```
            self.save_model()

        return True

    def printMetrics(self):
        print("MAE: " + str(self.metricMAE.get()))
        print("RMSE: " + str(self.metricRMSE.get()))
        print("R2: " + str(self.metricR2.get()))

    def prediction(self, entryData):
        return self.model.predict_one(entryData)

    def save_model(self):
        # by writing to tmp file operation becames atomic
        tmp = "tmp.plk"
        with open(tmp, 'wb') as f:
            pickle.dump(self,f)
        os.replace(tmp, self.modelName)

    def load_model(self):
        with open(self.modelName, 'rb') as f:
            return pickle.load(f)

def parse_response_data(inputData):
    inputData = inputData[2:]
    inputData = inputData[:-3]
    data_lst = inputData.split(",")

    data = {
        "age":int(data_lst[0]),
        "sex":data_lst[1],
        "bmi":float(data_lst[2]),
        "children":int(data_lst[3]),
        "smoker":data_lst[4],
        "region":data_lst[5]
    }

    # target are the charges int and float due to logistic regression
```

```
    return {"data": data, "target": float(data_lst[6])}
```

Listing 8: `kafkaConsumer.py`

```
from executeCmd import execute_cmd

KAFKA_PATH = "~/KAFKA/bin/"

def create_topic(server_addr, replication_factor, partitions, topic_name):
    """
    creates topic
    usage example params ca.alpha.olypus.pt 1 3 health_data1
    """
    cmd = KAFKA_PATH + "kafka-topics.sh --create --bootstrap-server " + str(server_addr) + ":9092 --replication-factor " + str(replication_factor) + " --partitions " + str(partitions) + " --topic " + str(topic_name)
    execute_cmd(cmd)
    return

def delete_topic(server_addr, topic_name):
    """
    creates topic
    usage example params ca.alpha.olypus.pt 1 3 health_data1
    """
    cmd = KAFKA_PATH + "kafka-topics.sh --bootstrap-server " + str(server_addr) + ":9092 --delete --topic " + str(topic_name)
    execute_cmd(cmd)
    return

def receive_data(server_addr, topic_name, max_messages):
    cmd = KAFKA_PATH + "kafka-console-consumer.sh --bootstrap-server " + str(server_addr) + ":9092 --topic " + str(topic_name) + " --from-beginning --max-messages " + str(max_messages)
    return execute_cmd(cmd)

def receive_data_offset(server_addr, topic_name, partition, offset):
    cmd = KAFKA_PATH + "kafka-console-consumer.sh --bootstrap-server " + str(
```

```
server_addr) + ":9092 --topic " + str(topic_name) + " --max-messages 1 --partition " + str(partition) + " --offset " + str(offset)
    return execute_cmd(cmd)
```

Listing 9: `consumerController.py`

```
import kafkaConsumer as kafka
import iml
import signal
from executeCmd import execute_cmd

KAFKA_TOPIC = "trainer"
KAFKA_PREDICTION_TOPIC = "analyzer"
KAFKA_ADDR = "localhost"
KAFKA_PARTITIONS = "1"
KAFKA_REPLICATION = "1"

def init():
    # Create Learn Data Topic
    kafka.create_topic(KAFKA_ADDR, KAFKA_REPLICATION, KAFKA_PARTITIONS, KAFKA_TOPIC)
    # Create Prediction Data Request Topic
    kafka.create_topic(KAFKA_ADDR, KAFKA_REPLICATION, KAFKA_PARTITIONS, KAFKA_PREDICTION_TOPIC)
    # Init Model
    global learner
    learner = iml.Iml()
    print('INFO: Init Data Topic and Model Sucess')
    return True

def clean():
    # Delete Learn Data Topic
    print("INFO: Deleting " + KAFKA_TOPIC + " topic...")
    kafka.delete_topic(KAFKA_ADDR, KAFKA_TOPIC)
    # Delete Prediction Data Request Topic
    print("INFO: Deleting " + KAFKA_PREDICTION_TOPIC + " topic...")
    kafka.delete_topic(KAFKA_ADDR, KAFKA_PREDICTION_TOPIC)
    # Delete Model
```

```
    print("INFO: Topics Deleted. Deleting Model...")
    cmd = "rm model.plk"
    execute_cmd(cmd)
    print('INFO: Cleaning Sucess')
    return True

def handler(signum, frame):
    global flag
    res = input("ACTION: Ctrl-c was pressed. Do you really want to exit? y/n
        ")
    if res == 'y':
        flag = False

def continuous_analysis(offset, partition):
    global flag
    flag = True
    signal.signal(signal.SIGINT, handler)
    global learner
    print('INFO: Continuous Analysis Started. Waiting...')
    # init: data needs to be loaded to access by offset atfer
    kafka.receive_data(KAFKA_ADDR, KAFKA_TOPIC, offset + 2)
    while(flag):
        print('\nINFO: Waiting Data From Kakfa')
        response = str(kafka.receive_data_offset(KAFKA_ADDR, KAFKA_TOPIC,
            partition, offset))
        offset += 1
        print('INFO: Data Received From Kakfa')
        if(response != "b\'\'"):
            data = iml.parse_response_data(response)
            print('INFO: Analysing Data With Incremental Learning Model')
            if (not learner.learn(data["data"], data["target"])):
                print('ERR: Incremental Learning Model Did Not Learn')
            print('INFO: Data Analysed')
        else:
            print("ERR: Kafka Response Empty")
            return
    print("INFO: Saving Model...")
    learner.save_model()
```

```
    print('INFO: Analysis Stopped\n')
    return offset

def predictions_request(offset, partition):
    global flag
    flag = True
    signal.signal(signal.SIGINT, handler)
    print('INFO: Loading Model')
    predictor = iml.Iml().load_model()
    print('INFO: Waiting Prediction Data From Kakfa')
    kafka.receive_data(KAFKA_ADDR, KAFKA_PREDICTION_TOPIC, offset + 2)
    while(flag):
        print('\nINFO: Getting Data From Kakfa')
        response = str(kafka.receive_data_offset(KAFKA_ADDR,
            KAFKA_PREDICTION_TOPIC, partition, offset))
        print('INFO: Data Received From Kakfa')
        offset += 1
        if(response != "b\'\'"):
            data = iml.parse_response_data(response)
            print('INFO: Predicting from Data With Incremental Learning Model
                ')
            prediction_result = predictor.prediction(data["data"])
            print("INFO: Prediction Result of Data " + str(prediction_result)
                 + " \n From Data: " + str(data))
        else:
            print("ERR: Kafka Response Empty")
            return
    predictor.printMetrics()
    return offset
```

Listing 10: `createTestFiles.py`

```
import os

def delete_files():
    os.system("rm -r ./testFiles")

def generate_files(quantity, nbytes):
```

```
    os.system("mkdir ./testFiles")
    for i in range(quantity):
        filename = "./testFiles/testFile_" + str(i)
        with open('%s'%filename, 'wb') as f:
            f.write(os.urandom(nbytes))
```

Listing 11: `timeProgram.py`

```
import timeit
from time import sleep

def start():
    global start
    start = timeit.default_timer()

def end():
    global start
    stop = timeit.default_timer()
    result = stop - start
    return result

def delay(event):
    print("INFO: Sleeping after " + event + "..." , end =" ")
    sleep(1)
    print("Waking Up!")
```