UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

LUCAS CARDOSO TAVARES

A Blockchain-Based Platform for Federated Learning

Work presented in partial fulfillment of the requirements for the degree of Bachelor in Computer Engineering

Advisor: Prof. Dr.
Claudio Fernando Resing Geyer
Coadvisor: Prof. Dr.
Julio Cesar Santos dos Anjos

Porto Alegre
October 2022

ACKNOWLEDGMENTS

Agradeço primeiramente a minha família. Especialmente a minha mãe, Márcia Helena De Moura Cardoso, meu pai, José Amado Azambuja Tavares e meus irmãos, Kauê Cardoso e Fábio Cardoso Tavares os quais foram a principal força motriz em todo meu processo de desenvolvimento. Sempre me apoiaram e me motivaram durante toda a jornada da graduação, sempre me entendendo e nunca me deixando desamparado. Sem vocês nada disso seria possível.

Agradeço também aos meus colegas, que tornaram o caminho da graduação mais fácil e agradável. Que compartilharam lado a lado comigo as frustrações, superações e realizações que é a árdua batalha da graduação.

Agradeço aos meus amigos, que sempre compreenderam os motivos das inúmeras ausência e sempre me apoiaram durante todo esse processo.

Agradeço aos meus professores, que me ensinaram a complexa arte de pensar e resolver problemas. E que, acima de tudo, sempre se mostraram dispostos a disseminar conhecimento e ajudar nesse duro processo que é aprender.

Agradeço em especial ao meu orientador Prof. Dr.Claudio R. Geyer por toda ajuda e dúvidas sanadas, ao Dr. Julio Cesar Santos dos Anjos pela coorientação, dica e suporte durante a elaboração desse trabalho.

ABSTRACT

Federated learning is an emerging AI technique that removes the need to centralize data on a server. In it, a server distributes machine learning tasks to the edge nodes, which perform the tasks on their data and send the results back to the server (aka. an update). This requires the development of a negotiation system between nodes that want to request machine learning task execution and nodes that want to sell their computation power in exchange for a reward. Some studies have used Blockchain in conjunction with federated learning to establish a secure, decentralized, and fair trade environment with a bargaining chip for these machine learning tasks. A key issue with the studies is that none of them focus on analyzing the gas fee spent by network transactions, and today, it is known that the main blockchain networks that guarantee security and privacy properties are public blockchains, with high transaction costs and low TPS. This work presented a trading platform in which edge nodes realize machine learning tasks previously agreed upon with a Requester and, after completing them, transmit the results back in exchange for a digital currency. The platform features a data matching system, sending offers and job contracts. The platform's major goal is to combine the security and anonymity of a public blockchain while providing higher scalability in transaction fee levels. To test the solution feasibility, the platform's basic functionalities were developed, and an environment with four nodes was prototyped running a Convolution Matrix filter in a federated approach. The number of interactions and fees paid were evaluated and compared to similar research studies.

Keywords: Blockchain, Federate Learning, Big-data, Trading, DAO.

A Blockchain-Based Platform for Federated Learning

RESUMO

O aprendizado federado é uma técnica de IA emergente, que remove a necessidade da centralização dos dados em um servidor. Nela, um servidor envia tarefas de aprendizado de máquina para os *edge nodes*, que então executam a tarefa sobre seus dados e enviam apenas o resultado de volta para o servidor, um *update*. Para isso é necessário criar-se um sistema de negociação entre *edge nodes*, que queiram executar tarefas de machine learning, e *edge nodes*, que queiram vender seu poder computacional em troca de alguma recompensa. Nesse contexto, alguns estudos vêm aplicando Blockchain junto ao aprendizado federado em busca de conseguir criar um ambiente de negociação seguro, descentralizado, e justo com alguma moeda de troca para que essas tarefas de aprendizado de máquina sejam transacionadas. Um problema pivotal entre os estudos é que nenhum foca em analisar o gasto em taxa de gas por transações na rede, e hoje, sabe-se, que as principais redes blockchain que garantem as propriedades de segurança e privacidade são blockchains públicas e, por sua vez, com alto custo de transação e baixo TPS. Nesse trabalho, é proposta uma plataforma de negociação onde *edge nodes* executam tarefas previamente acordadas com um Requisitante e, após executá-las, envia o resultado de volta em troca de uma moeda digital. A plataforma apresenta um sistema de correspondência de dados, de envio de ofertas e contratos de trabalho. O principal objetivo da plataforma é unir a segurança e privacidade de uma blockchain pública garantindo uma maior escalabilidade em níveis de taxa por transação. Para verificar a viabilidade da solução foram implementadas as funcionalidades básicas da plataforma e prototipado um ambiente com 4 nodos executando um filtro de Matriz de Convolução de forma federada. O número de interações e os valores gastos em taxas foram analisadas e comparados com soluções estudadas semelhantes.

Palavras-chave: Blockchain, Federated Learning, Big-Data, Trading, DAO.

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
FL	Federated Learning
ML	Machine Learning
IGs	Interest Groups
POCI	Proof of Common Interest
IPFS	InterPlanetary File System
DAO	Decentralized Autonomous Organization
PoW	Proof of Work
PoS	Proof of Stake
POCI	Proof of Common Interest
CID	Content Identifier
IoT	Internet of Things
TPS	Transactions Per Seconds
ZK	Zero Knowledge
TVL	Total Value Locked
ICO	Initial Coin Offering

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Big data is a crucial asset in the scientific and economic world (ZHENG et al., 2020), it has been applied in almost every area, including retail, healthcare, financial services, government, agriculture, and customer service (RABAH, 2018). It is a strong tool that is helpful to any firm that wants to integrate data to solve nagging questions about its business. Overall, the need for big data is widespread across all industries and businesses. Big data explosive growth has become a natural process due to its vast applicability, and the endeavor to secure every part of it has become a hard task. (DOKU; RAWAT; LIU, 2019). Another major issue is the centralized manner in which big data is currently stored, the majority of it is monopolized by GAFA (Google, Apple, Facebook, and Amazon) and the Chinese big techs such as Alibaba, Tencent, Baidu, and TikTok (RABAH, 2018; DOKU; RAWAT; LIU, 2019).

Due to a lack of competition that exists in a monopoly, this not only raises market control and privacy concerns but also presents a single point of failure issues in the architecture (DOKU; RAWAT; LIU, 2019; BEATTIE, 2021). In this context, blockchain appears as a great ally due to its decentralization, tamper-resistant, and tamper-evident properties (YAGA et al., 2018). It not only mitigates single-point failure through distributed consensus but also includes incentive mechanisms to encourage participants to collaborate and effectively contribute to the system without the need for a centralizing entity (MA et al., 2020).

The emergence of the Internet of Things (IoT) and social networking apps, among other things, has resulted in an exponential increase in the generation of data for machine learning at multiple decentralized edge nodes (WANG et al., 2018). Traditional machine learning systems, which transmit all data from edge nodes to a central server, not only lose efficiency but also have significant risks of privacy leakage and information hijacking (MA et al., 2020). To address these problems, a distributed machine learning approach called Federated Learning (FL) emerges. The main advantage of Federated Learning is that it decouples the model training from the necessity of direct access to the training data (MCMAHAN et al., 2016).

While Federated Learning guarantees privacy, it still has some security, and privacy issues (HOU et al., 2021). In this context, blockchain appears as a great ally, it can ensure secure interaction in an untrusted environment by enabling the record of transactions in a distributed and shared ledger (YAGA et al., 2018). Apart from that, some

blockchains provide a native digital currency which is a suitable and flexible incentive mechanism that can be used to reward devices that use their computing resources to perform machine learning tasks (YUE et al., 2017).

Making federated learning happen in a decentralized, fair, efficient, and secure environment remains a huge challenge. Although some research integrating blockchain to federated learning has yielded significant results, few have focused on how to make this union a gas fee scalable, secure, and efficient way. The main objective of this work is to present a trading platform for machine learning tasks with mainly focus on gas fee, autonomy, security, and privacy. These properties are accomplished by the use of a Layer-2 Roll-up running on top of a public Blockchain, all models are stored using IPFS and the organization is organized by a DAO.

The remainder of this work is organized as follows. First, Chapter 2 introduces some essential concepts for the understanding of this work, then, Chapter 3, is made a review of some related works. Chapter 4 introduces the proposed system model of this work, and in Chapter 5 first describes the methodology and prototyping, then the evaluation and results of the model. Finally, in Chapter 6 is discussed the conclusions and future works of this dissertation.

2 BACKGROUND

This Chapter introduces some background concepts used as the underlying of this work. The Chapter starts by explaining about Blockchain 2.1, then goes through Federated Learning 2.2, DAO 2.3, Layer 2 2.4, IPFS 2.5, and finally, Section 2.6 introduces both blockchain solutions used in this work.

2.1 Blockchain

Blockchain is a distributed digital ledger that organizes cryptographically signed transactions into blocks. After validation and consensus, each block is cryptographically connected to the preceding one (making it tamper obvious). Older blocks become increasingly difficult to change when new blocks are added (creating tamper resistance) (DOKU; RAWAT; LIU, 2019). The immutability and transparency of blockchain technology, in particular, assist in eliminating human mistakes and the need for manual intervention due to conflicting data (DINH et al., 2018). There are typically two categories of blockchain, private blockchain and public blockchain. Public blockchains are decentralized ledgers open to anyone publishing blocks. Private blockchains are more centralized and the user must have permission to publish a block (SALIMITARI; CHATTERJEE, 2018).

The seven core components of a blockchain are: cryptographic hash function, transaction, asymmetric-key, addresses, ledger, block, and consensus models. Each one of them addresses a specific blockchain functionality and could have a different type of implementation according to the main necessities of the system (YAGA et al., 2018). One of the key components of a blockchain, and the one that will be more deeply addressed here, is the consensus model. Consensus model is the mechanism used to determine which user will publish the next block. The most common consensus models are Proof of Work (PoW) and Proof of Stake (PoS).

Proof of Work gives the right to publish the next block to the first user that solves a high expensive computationally puzzle, usually, a concurrency amount is given to the winner as a reward for the work done. The puzzle is created in such a way that solving it is tough but checking a solution is simple. The biggest drawback of PoW is that to solve the puzzle a lot of energy, in fact, Bitcoin uses 707 kilowatt-hours (kWh) of electricity per transaction, while Ethereum [1] 64 kWh per transaction (POWELL, 2022) both of them

[1] As I write this work, Ethereum is working on making its proof-of-work to proof-of-stake migration.

are PoW. The typical process of a transaction(in PoW blockchain is shown in Fig. 2.1, adapted from (EUROMONEY, 2020).

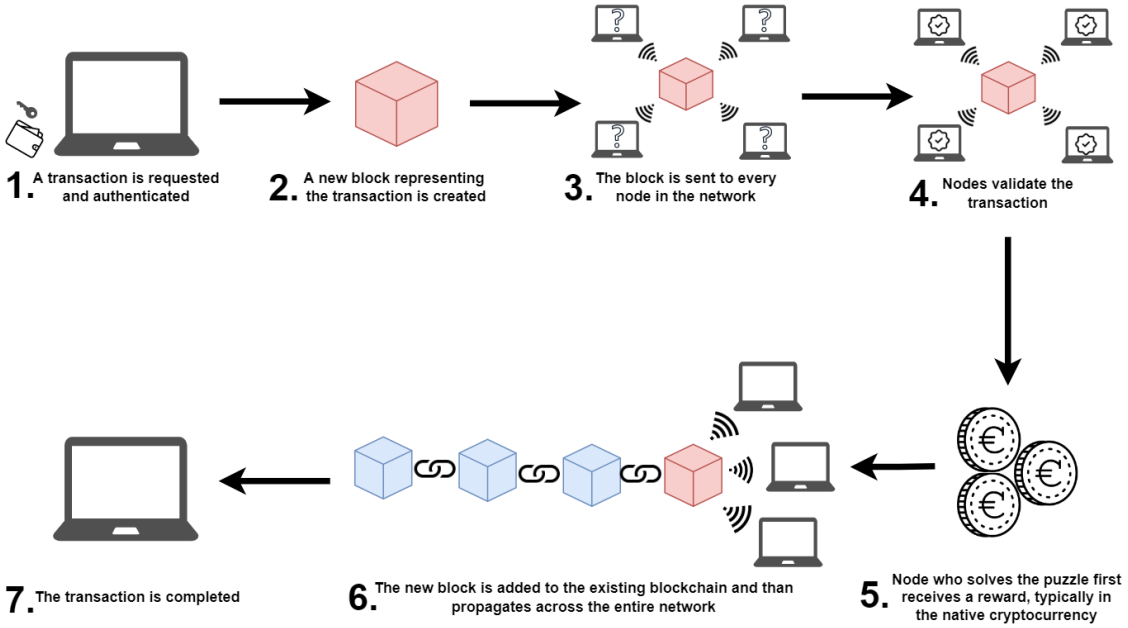

Figure 2.1 – Typically process of a transaction in PoW

On the other hand, Proof of Stake gives the right to publish the next block based on the ratio overall stake amount of users, often an amount of cryptocurrency. PoS is based on the belief that the greater an user stake in the system, the more likely they are to desire the system to succeed (DINH et al., 2018). In PoS, different of PoW, the reward for validate a block is not given to only one user, usually, it is divided between the validators based on their stake amount. The major benefice of PoS when compared with PoW is that it does not require energy to validate a block, making it more energy efficient and eco-friendly. The major drawback of PoS, when compared with PoW, is that validators with large holdings can have excessive influence on transaction verification, making the network more centralized (DALY, 2022).

2.2 Federated Learning

Federated Learning (FL) is a decentralized machine learning technique that allows edge devices to train a shared prediction model collaboratively while maintaining all training data on the device, decoupling machine learning from the obligation to store data on the cloud/server (MCMAHAN, 2017). It is used to train a global model by the use of

All tests and results in this work were built over Ethereum PoW.

multiple edge devices while protecting their privacy. In it, multiple devices are scattered to execute machine learning tasks locally on their own data set, generating local updates and then submitting them to a central server that aggregates the results and updates the global model parameters. A representation of the usual steps done in a Federated Learning process is shown in Fig. 2.2, adapted from (WIKIPEDIA, 2022).

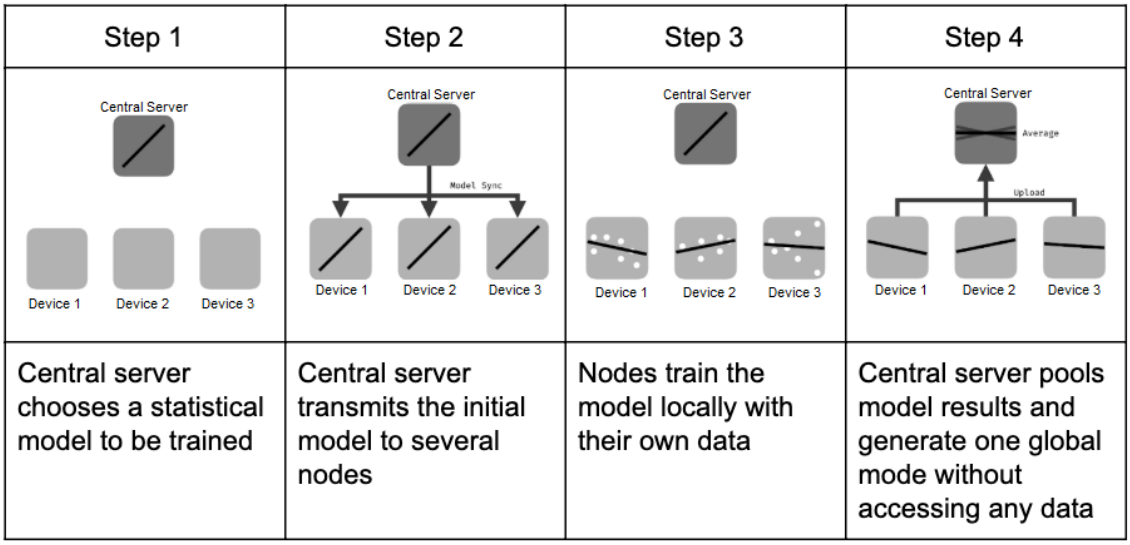

Figure 2.2 – Federated Learning process steps.

2.3 DAO - Decentralized Autonomous Organization

A Decentralized Autonomous Organization (DAO) is a non-centralized entity that makes decisions based on a set of rules that are enforced on smart contracts inside a blockchain (COINTELEGRAPH, 2022). A DAO is operated by a set of smart contracts[2], these contracts define the rules of the organization, hold the membership treasury, are very straightforward, easy to verify, and open to public auditing. In this way, every member is able to completely comprehend how the organization works at every stage (SHUTTLEWORTH, 2021).

When opposed to traditional organizations, the DAO's major benefit is that it has democratized organizations, and any modifications must be approved by a majority of the members rather than being carried out by a single party (ETHEREUM, 2022a). A comparison between DAOs and traditional organizations is shown in Table 2.1.

[2]Smart contracts are digital contracts stored on a blockchain that are automatically executed when predetermined terms and conditions are met

Property	DAO	Traditional Organization
Organization	Typically straight and completely democratic	Most often hierarchical.
Governance	Governance based on community	Governance is mostly based on executives, Board of Directors, activist investors
Operation	Operates by public smart contracts	Operates by owners' rules
Validation	Interactions are recorded on the blockchain, making it easy to be validated/verified by any member.	Interactions may or may not be recorded, and if recorded, only some members have permission to validate/verify.
Changes	For any changes to be put into effect, members must vote	Depend on the arrangement, only one party may be required to make changes, or voting may be proposed
Voting	Without a reliable middleman, votes are counted and the result is implemented	If voting is permitted, the results of the vote must be handled manually. Votes are counted internally
Management	The given services are managed automatically and decentralized (for example distribution of philanthropic funds)	Require manual handling or centrally managed automation and are easily manipulable
Transparency	Every action is open to the public and transparent.	The public is often excluded from the activity and it is private

Table 2.1 – DAO x Traditional Organization comparative

2.4 Layer 2

A Layer 1 network, also named Mainnet, is the underlying infrastructure of a blockchain. It not only defines the ecosystem's core rules but also validates and registers the transactions. Most public Layer 1, such as Ethereum and Bitcoin, have an architecture with a greater emphasis on decentralization and security and, because of that, they end up lacking scalability. This is known as the Scalability Trilemma, shown in Fig 2.3. The Scalability Trilemma states that the scalability trade-offs are unavoidable between three fundamental properties: decentralization, scalability, and security, in such a way that we can only have two out of either decentralization, security, or scalability. Because of it achieving a network with security over a broadly decentralized network while managing a blazing throughput is the holy grail of blockchain technology.

One of the ways to try to solve the Scalability Trilemma is with the use of Layer 2. The most popular example is the *Lightning Network*[3] used on top of Bitcoin to improve the transaction speed (STAFF, 2022). With Lightning Network, Bitcoin is capable of processing 1,000,000 transactions per second (TPS), while without it can only process about

[3]The Lightning Network is a "layer 2" payment protocol layered on top of Bitcoin. It is intended to enable fast transactions among participating nodes and has been proposed as a solution to the bitcoin scalability problem.

7 transactions per second (BITPAY, 2022).

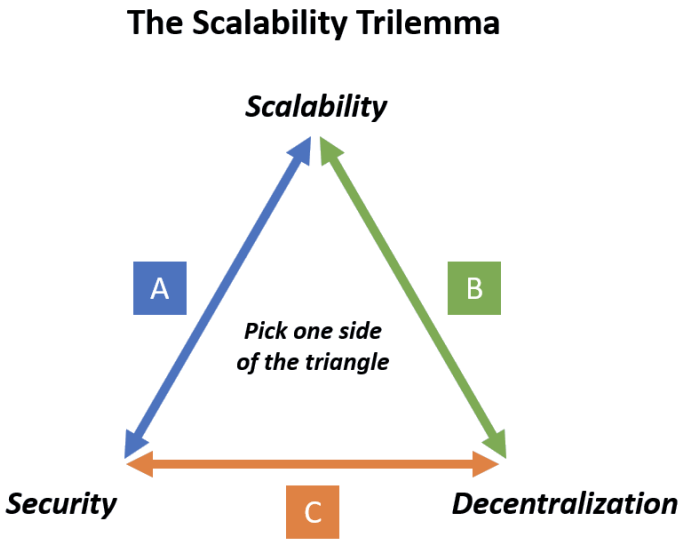

Figure 2.3 – Scalability Trilemma. Source: (ZOCHOWSKI, 2018)

On the other hand, Layer 2 refers to the protocol that runs on top of an existing blockchain (Layer 1) with the main goal of increasing scalability, transaction speed, and efficiency of the network. The core concept behind Layer 2 is to provide a framework that conducts transactions off-chain, lowering the strain on the blockchain and obtaining faster transaction speeds (SGUANCI; SPATAFORA; VERGANI, 2021). The most popular Layer 2 solutions approaches are State Channel, Side-chains, and Rollups (Optimistic-Rollup and ZK-Rollup). The remainder of this Section describes each of the solutions in detail, explaining their strengths and weaknesses. A summary of their trade-offs can be seen in Table 2.2, adapted from (ZOCHOWSKI, 2021). As we can see, there is no best solution, but the one that best applies is dependent on the system constraints.

		State Channels	Sidechain	Optimistic-Rollup	ZK-Rollup
User Experience	Scaling Capability	infinite	excellent	weak	general
	Cost of Transaction	approx. 0	low	low	low
Security	Security off-chain	weak	weak	strong	extremely strong
	Security on-chain	no	no	yes	yes
Usability	Coding difficulty	easy	easy	medium	hard
	Applicability	weak	strong	medium	medium

Table 2.2 – Layer 2 solutions trade-off

2.4.1 State Channel

State Channels allow users to do many transactions off-chain while only submitting two transactions to the bottom layer, one to open the channel and another to close the channel, resulting in extremely high throughput at a minimal cost (MONOLITH, 2021). It is a basic peer-to-peer protocol that allows two parties to carry-out several transactions with another while only posting the final results to the blockchain. The channel employs cryptography to prove that the summary data it generates is the product of a legitimate sequence of intermediate transactions, and a "multisig" smart contract guarantees that the proper parties sign the transactions (ETHEREUM, 2022a). The most popular use case of State Channels is Payment Channels, and it is shown in Fig 2.4, adapted from (CSIRO, 2022).

While State Channels provide low-cost payments, the expenses of putting up the on-chain contract on Mainnet during the launching phase might be prohibitively expensive, particularly when gas prices rise. Another drawback of using State Channels is that it assumes the full availability of both peers to constantly check the activity and contest challenges when necessary (ZOCHOWSKI, 2021). Also, only the final outcome of the transaction and the final account status are registered on the main chain; none of the intermediate transactions are. Another drawback of State Channel is modifiability, a new wallet or extension to the existing wallet is needed to support the micropayment protocol.

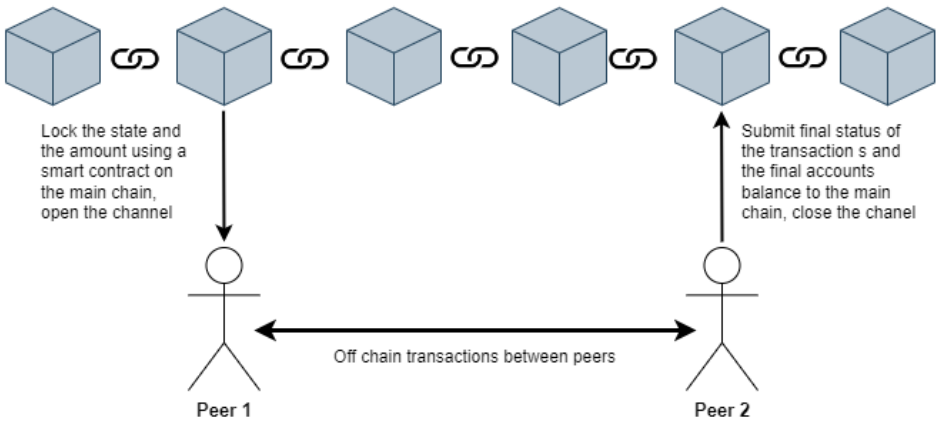

Figure 2.4 – Payment Channel

2.4.2 Side-Chains

Side-Chains work in a separate network with a self-contained consensus mechanism that can be optimized for speed and scalability, running separately from the Mainnet, and operating independently. Side-Chains are connected with the Mainnet via a two-way bridge and to send and use it the user must lock their coins in a smart contract on the main layer and get credits in the side-chain address (MONOLITH, 2021). Once the user has credits in the side-chain, transactions can occur off the main chain and, in the end, just the final account balance is submitted to the main chain. A representation of a Side-Chain is shown in Fig 2.5, adapted from (MONOLITH, 2021).

Depending on the consensus mechanism used in the Side-Chain the transaction per second (TPS) can increase and the transaction fees reduce significantly. Nonetheless, the drawback of it is too high, Side-Chains trade off a high degree of decentralization and trustfulness for scalability (BUTERIN, 2021). Apart from that, it uses its own consensus mechanism, losing the benefits of the bottom layer consensus like security and requiring higher trust assumptions over the users and validators.

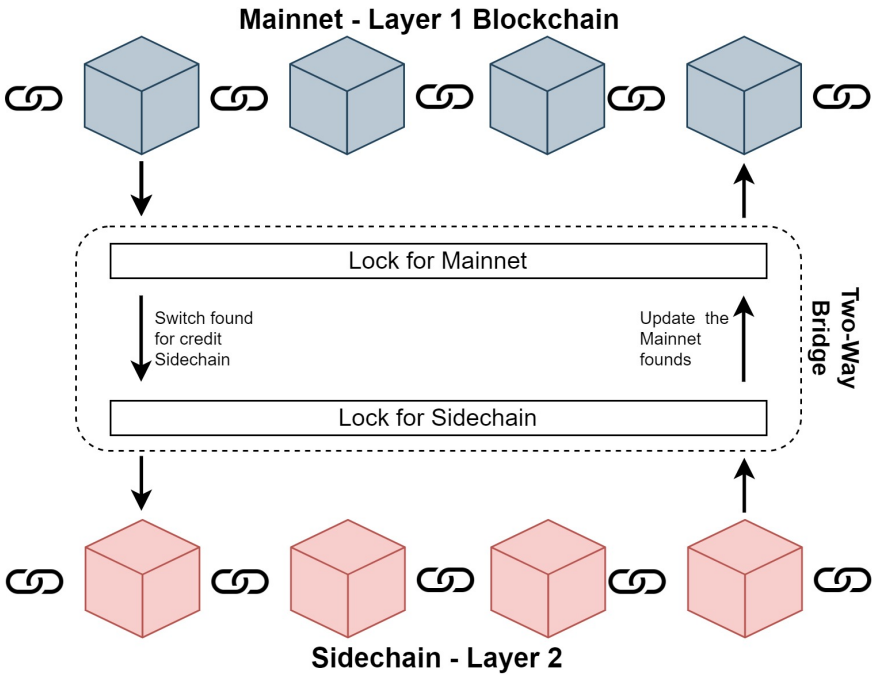

Figure 2.5 – Side-Chain

2.4.3 Rollup

Rollups perform transaction execution outside Mainnet and then compress these transactions and post them to Mainnet where consensus is reached. Rollup is the name given to the entire process of executing transactions, taking the data, compressing it, and rolling it up to the main chain in a single batch. As transaction data is posted in the bottom layer, this allows Rollups to still fully rely on the security of Mainnet (BUTERIN, 2021). Rollup tries to be the holy grail of scaling as it allows for deploying all of the existing smart contracts present on Mainnet to a Rollup with no changes while relying on the bottom layer security.

The way the Mainnet knows that the posted data is valid depends on the specific Rollup implementation. Usually, each Rollup implementation deploys a set of smart contracts on Mainnet responsible for processing deposits and withdrawals and verifying proof (JAKUB, 2021). The verifying proof is the major difference between each Rollup implementation, and the most known type of implementations are **Optimistic-Rollup** and **ZK-Rollup**. A representation of Layer 2 Rollup is shown in Fig. 2.6, adapted from (BUTERIN, 2021). Table 2.3 shows a summarized comparison between the Optimistic-Rollup and ZK-Rollup.

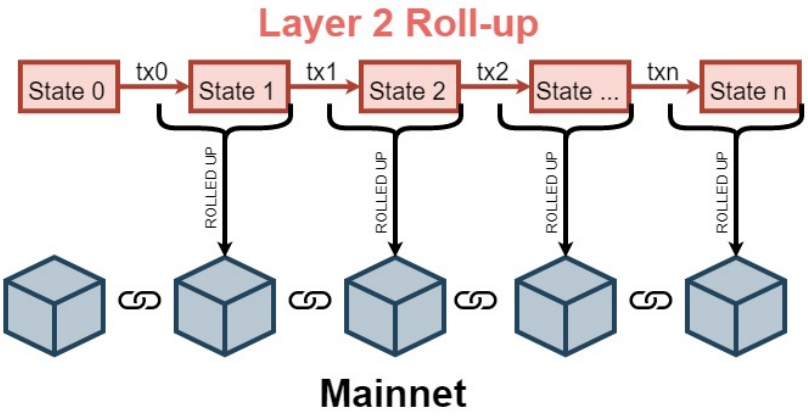

Figure 2.6 – Rollup

1) Optimistic-Rollup: This approach assumes the authenticity of off-chain transactions and does not publish any proof of the legitimacy of transaction batches that are put on-chain. To identify instances where transactions are submitted by a malicious actor, this method depends on fraud-proving (BUTERIN, 2021). The network members have a time window after a batch of transactions is published on the Mainnet to contest the batch's authenticity by providing a fraud-proof. Upon the submission of a fraud-proof, the system

Property	Optimistic Rollups	ZK Rollups
Fixed gas cost per batch	Aprox. 40,000 (a lightweight transaction that mainly just changes the value of the state root)	Aprox. 500,000 (verification of a ZK is quite computationally intensive)
Withdrawal period	1̃ week (withdrawals need to be delayed to give time for someone to publish a fraud-proof and cancel the withdrawal if it is fraudulent)	Very fast (just wait for the next batch)
Complexity of technology	Low	High (ZK are very new and mathematically complex technology)
Generalizability	Easier (general-purpose EVM Rollups are already close to Mainnet)	Harder (ZK proving general-purpose EVM execution is much harder than proving simple computations)
Per-transaction on-chain gas costs	Higher	Lower (if data in a transaction is only used to verify, and not to cause state changes, then this data can be left out, whereas in an Optimistic Rollup it would need to be published in case it needs to be checked in a fraud-proof)
Off-chain computation costs	Lower (though there is more need for many full nodes to redo the computation)	Higher (Zero-Knowledge proving especially for general-purpose computation can be expensive, potentially many thousands of times more expensive than running the computation directly)

Table 2.3 – Optimistic Rollups x ZK Rollups Source: (BUTERIN, 2021)

initiates a dispute resolution process in which the suspicious batch of transactions is once more carried out, but this time on the Mainnet. The sender of this transaction is penalized if the execution demonstrates that the transactions are fraudulent.

To solve a dispute resolution Optimistic-Rollup must have a mechanism that can replay a transaction with the identical state that existed when the transaction was first conducted on the Rollup in order to be able to execute the batch on Mainnet. One of the trickier aspects of Optimistic Rollup is often accomplished by establishing a separate manager contract that substitutes the state from the Rollup for particular function calls (AL, 2021). Entering a dispute resolution process is time expensive and it should be an exception situation, otherwise, it could generate a lot of overhead.

2) ZK-Rollup: Zero-Knowledge Rollups runs computations off-chain like Optimistic Rollup but different from it ZK-Rollup submits the compressed batch with a validity cryptographic proof to the chain. There is no dispute resolution at all, in this approach, every batch submitted to Mainnet includes a cryptographic proof called Zero-Knowledge. This proof can be easily verified by the Mainnet, and if a fraudulent batch is detected it can be rejected straight away (AL, 2021).

The major drawback of ZK-Rollups is due to the complexity of the Zero-Knowledge technology. This complexity not only makes it more difficult to develop a compatible ZK Rollup with the Mainnet but also requires a high hardware requirement to compute ZK proofs, which may encourage centralized control of the chain and/or reduce the number of participants that can submit batches to the Mainnet (ETHEREUM, 2022a).

2.5 IPFS - Inter Planetary File System

Despite the Internet's widespread use, its data storage remains largely centralized, whether it be physically or virtually, on servers or cloud computing platforms. Most parts of these servers are controlled by big companies like Google, Amazon, Microsoft, Facebook, and IBM (MUSHARRAF, 2021). This kind of centralization can cause a number of issues, such as what happened on October 4, 2021, when Facebook, Instagram, and WhatsApp experienced an extended global outage that lasted for hours (ISAAC; FRENKEL, 2021). Where you suddenly lost access to your Facebook and Instagram messages and posts, as well as your ability to download WhatsApp-sent pictures. That was due to the "location"—or, to put it another way, the server—where all the data was kept being inaccessible.

In this context, the InterPlanetary File System (IPFS) appears as a great solution. IPFS is a distributed, peer-to-peer system for storing and accessing files, websites, applications, and data, where contents are accessible through many peers located around the world (STAFF, 2021). This not only makes the data storage decentralized but also leverages the privacy and security of the data since it uses encrypted Content Identifiers (CIDs) to store the data. A content identifier, or CID, is a label used to point to material in IPFS. It doesn't indicate where the content is stored, but it forms a kind of address based on the content itself. CIDs are short, regardless of the size of their underlying content.

2.6 Ethereum & Arbitrum

In this Section are presented, Ethereum and Arbitrum, the selected blockchains to compose the structure of the proposed model. First will be described Ethereum, the public blockchain used as Layer 1 then Arbitrum, a Layer 2 Optimistic Rollup solution.

1) Ethereum: is a public blockchain that had it is whitepaper written in Novem-

ber of 2013 by *Vitalik Buterin*, and had it is ICO[4] in July 2014, is the first blockchain with smart contract functionality, allowing developers to build Dapps[5]. Nowadays, the Ethereum network is the most popular decentralized public blockchain with smart contracts functionality available, it is a proof of work blockchain and has Ether as a native cryptocurrency. Ethereum is already used for huge companies like Toyota, Samsung, Microsoft, Intel, and J.P. Morgan, also, it is the second-biggest blockchain with more market cap (COINMARKETCAP, 2022), losing only to Bitcoin, which has no smart contracts available.

Apart from that, Ethereum is the fastest growing network with the largest active community, with more than 4,000 active developers and with over 900 commits per week (VENTURES, 2021; CHOY, 2022). As this piece is written, Ethereum is working on making its proof-of-work to proof-of-stake migration, which means it will no longer need energy-intensive mining and instead secures the network using staked Ether. This migration is called "The Merge" and is set to happen on September 13, 2022. It will bring more scalability, security, and sustainability to the network and make it ESG[6]-friendly (ETHEREUM, 2022b). Anyway, all tests and results in this work were built over Ethereum PoW.

2) Arbitrum: is a second-generation Layer 2 that provides higher throughput and more efficient dispute resolution than the normal Optimistic solutions. Arbitrum provides a complete Ethereum-compatible chain running smart contracts applications deployed in Ethereum Virtual Machine (EVM). It ensures the progress of Layer 2 and relays the safety of the Mainnet. Arbitrum approaches fraud-proof verification with a fine-combing method and focuses on a single point of transaction disagreement by employing multi-round fraud-proof (KALODNER et al., 2018), this translates to higher network performance. Similarly, because L2 transactions are not completed on L1, the gas block limit becomes insignificant. Nowadays, Arbitrum is the major Layer 2 used for Ethereum's DApps, being currently the Rollup project with the biggest total locked value, which means that have more credibility, and with transactions per second up to 40,000, and gas prices up to 100 times lower than Ethereum (NAMBIAMPURATH, 2022).

[4] An initial coin offering (ICO) is the cryptocurrency industry equivalent of an initial public offering (IPO).

[5] Decentralized applications, or dapps, are a revolutionary new approach of developing applications that use blockchain to eliminate centralized middlemen

[6] Environmental, social, and governance (ESG) criteria are used to screen investments based on corporate policies and encourage companies to act responsibly.

3 RELATED WORK

Many research activities concerning a decentralized data-sharing marketplace using blockchain as the underlying technology to improve privacy and security are being conducted. (YUE et al., 2017) designed a decentralized data-sharing model that protects the data owner's computation and output privacy through safe multiparty computing and differential privacy. However, his work focuses only on data sharing, dropping off issues such as dishonesty among participants, and evaluating the quality of the data. (JUNG et al., 2019) introduced *AccountTrades*, a set of large-data response protocols that create a secure environment through bookkeeping and accountability against dishonest users. The problem with this work is that it does not present a method to evaluate data quality and assumes that Brokers (who provide shopping services in general such as product listing, description, payment, delivery, etc...) are trustworthy, which leads to the system becoming more centralized and consequently to have less privacy.

(DAI et al., 2020) mitigate the existence of dishonest requesters and dishonest providers by allowing the consumers just to access the result of data analysis rather than the data itself. Although the system solves some honesty issues, it also 1) makes it impossible to negotiate the original data, 2) does not present a fair way to deal with providers/consumers who stop the action before the analysis finishes, and 3) does not present a way to evaluate the quality of data. (ZHENG et al., 2020) proposes a blockchain-based decentralized big data trading platform that keeps a summary and history of data evaluations made by users. It also ensures anonymity, autonomy, and data trading fairness. However, this system 1) cannot meet the need data trading requirements in the situation in which the data is only required for a limited time, 2) assumes that the requester's feedback on the data will always be honest, 3) only uses the Requester feedback as a metric to evaluate the data, and 4) has expensive smart contracts.

(SHAYAN et al., 2018) proposed a blockchain-based Federated Learning with fault tolerance, defense against some already known attacks, and scalability. However, it does not present any form of evaluating the quality of data, nor any incentive mechanism for the fair competition of heterogeneous nodes, just a payment reward. (DOKU; RAWAT; LIU, 2019) proposes an approach to determining the relevance of data based on the concepts of Interest Groups (IGs), and Proof of Common Interest (POCI), both proposed by the author. They also introduce the use of IPFS to make data storage more decentralized. However, this work does not provide the implementation and feasibility of

those proposed concepts.

(FAN et al., 2021) presents a trading system for FL, enabling heterogeneous nodes to use their computational capacity and local data to train models in exchange for a reward. The system improves network performance by using a hybrid blockchain-based model, where the consortium blockchain is responsible for resource trading and the public blockchain for payments. The models also ensure the meeting of the budget feasibility, RI, veracity, and computational efficiency by smart contracts implementing the DQDRA, a proposed reverse auction within the fixed budget to maximize the valuation of the requester. Finally, a smart contract incorporates a payment channel into the public blockchain, allowing for credible, rapid, low-cost, and high-frequency payment transactions between Requesters and edge nodes.

However, (FAN et al., 2021) not gives a mechanism to record service evaluations, and also the use of payment channels requires full peer availability and has high fee expensive in case of many disconnections between peers. Apart from that, the use of a private blockchain to run the auction not only makes the system more centralized and susceptible to malicious actors but also includes usability concerns, the user should be registered and interact with two blockchains.

Author & Year	**Approach**	**Strengths**	**Weaknesses**
(XIAO et al., 2019)	Introduction and evaluation of consensus algorithms against fault tolerance, performance, and vulnerabilities.	A good deal of consensus algorithms.	Mostly theoretical.
(DINH et al., 2018)	Proprietary benchmark to evaluate against latency and throughput.	Identifies performance bottlenecks. Test environment with 16 real nodes.	It only evaluates a few blockchains, and with few workloads.
(HUA et al., 2020)	Introduce a Federated Learning framework for intelligent control in Heavy Haul Railway.	Uses blockchain to improve security and privacy.	It does not use a reward technique and does not detail the blockchain network.
(DOKU; RAWAT; LIU, 2019)	Proposes an approach to determining whether data is relevant and storing that data in a decentralized way within an interest group.	It uses sharding to improve blockchain scalability and IPFS to store data on the network.	It is mostly theoretical and does not propose any form of reward.
(SALIMITARI; CHATTERJEE, 2018)	Compare different consensus algorithms within the context of IoT constraints.	Evaluate blockchains against accessibility, scalability, latency, throughput, and IoT suitability	It is mostly theoretical and does not evaluate against gas fee.
(HAN; GRAMOLI; XU, 2018)	Compare different popular blockchains, and discuss your suitability in an IoT environment.	Test environment with 2 up to 32 real nodes. Evaluate against latency and throughput.	Uses a very simple workload.
(ZHENG et al., 2020)	Propose a blockchain-based trading platform for big data.	Anonymity, confidentiality, and integrity. Stores previous evaluations to be used in the future.	Low TPS and high expensive gas fee.
(FAN et al., 2021)	Implements a trading system for services for Federated Learning using a Hybrid blockchain over an auction model.	Payment channel; Auction model as reward technique.	It requires full peer availability and does not store service evaluations. Also, the auction runs over a private blockchain.

Table 3.1 – Summary related work

Table 3.1 provides a brief comparison of the primary publications researched used

as the basis of this work, these papers are compared in terms of approach, strengths, and weaknesses. As we can see, most of the articles are completely theoretical, with only concepts and theories, lacking implementation and used technologies. This work not only shows and explains the model in Chapter 4, but also discusses the methodology and technologies used in Chapter 5.

Another weak point among the publications is concerning the autonomy of the system entities. In (FAN et al., 2021) work, neither the Requester nor the Data Provider has autonomy, since the smart contact is responsible for the entire process, from the selection of the winners to the closing deal. In (ZHENG et al., 2020) work, only the data provider has autonomy, the Requester has no autonomy to choose the data. The problem is that the entire selection process is done by an algorithm, leaving the Requester without autonomy during the process. In the proposed platform, the system is responsible for finding the best candidates for a given service, but the Requester has the autonomy to choose which, among the best, will be the winners and make them an offer, in turn, once the Trainer has received an offer, he has the autonomy to accept it or not.

Also, a pivotal problem among those studied is gas fees (transaction fee), in (FAN et al., 2021) work, a side-chain is used to make the auction, however, side-chains are extremely weak in terms of security due to the use of their own consensus (ZOCHOWSKI, 2021), and a payment channel is used to decrease the gas fees of the transactions, nonetheless, a payment-channel require integral availability of both peers. Also, if peers have many disconnections, many deploys should be done on the public network (to open the payment channel again) and the fee expense can be huge. In (ZHENG et al., 2020) work, no approaches to reduce the gas fees are used, and the entire system ran over a public blockchain, which makes the gas fees for each transaction not worth it.

This proposed platform uses a Layer 2 Rollup approach to reduce the gas fee while relaying the security of the Mainnet, a public blockchain. Apart from that, it is the only model that organizes the entire system using a DAO, and that ensures a high degree of autonomy and transparency. Table 3.2 summarizes the major differences between the proposed model and (FAN et al., 2021; ZHENG et al., 2020) works, which are the studied models with the highest degree of similarity, in autonomy, service evaluation, scalability, usability, data integrity, security, and the minimum number of interactions to close a deal.

	Proposed model	**(FAN et al., 2021)**	**(ZHENG et al., 2020)**
Autonomy	Requester can choose which of the winners he wants to make an offer to. Trainers can choose whether or not to accept an offer.	Data Provider can choose whether or not to sell their data to a Requester.	Service Provider can choose in to participate in an auction or not.
Service evaluation	Requester evaluates Trainer. Trainer evaluates Requester.	Requester rates the purchased data.	N/A
Scalability	Optimistic Rollups	Side-Chains	State Channels
Usability	All interactions are done through the DAO smart contract.	Registrations happen through a Key Generation Center. Interactions happen in two different blockchains.	User should register and interact with two blockchains. To make payments Requester must deploy a smart contract to open the Payment Channel.
Data integrity	IPFS	Encrypted	N/A
Security	Platform relays over public blockchain security and privacy.	Platform relays over public blockchain security and privacy.	Auction relays over private blockchain security. Payments are done in a public blockchain but using payment channels.
Min number of interactions to make a deal	3	5	4

Table 3.2 – Major differences between the proposed model Fan model and the Zheng model.

4 PROPOSED MODEL

This Chapter first shows an overview of the model architecture, then describes the entities that compose the system, explains the blockchain structure, and finally, discusses how the interactions with the system work.

4.1 Overview

An overview of the entire architecture is shown in Figure 4.1, where actors, represented by stick figures, are the entities who interact with the platform; DAO, represented by a paper, is the smart contract that organizes the entire platform, and where all actions start. DAO is deployed over Layer 2, and it is represented by the line connecting both; The Rollup process occurs internally on Layer 2, once it is done the transactions are stored on Layer 1, and it is represented by the dotted line. Table 4.1 shows the notations within this paper.

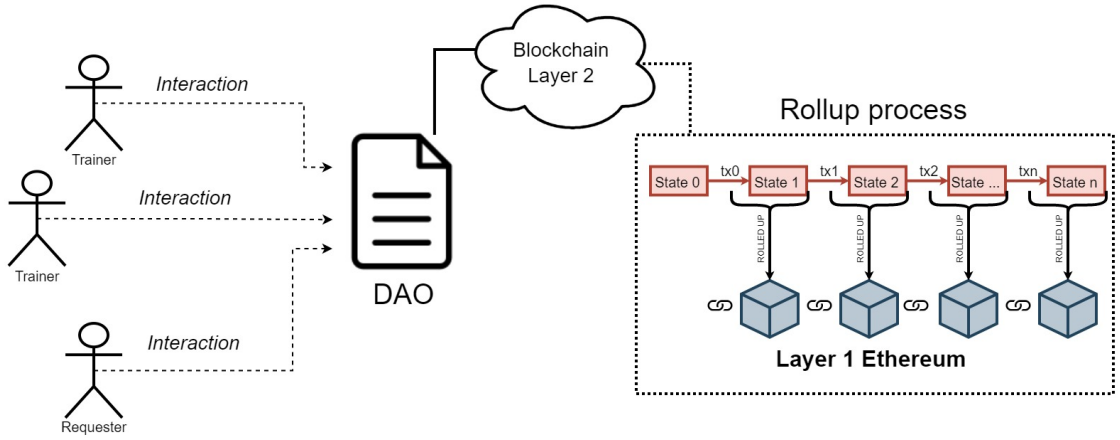

Figure 4.1 – Model Architecture Overview

Table 4.1 – Notation of Explanation

Notation	Explanation
$T = \{T_1, ..., T_i, ...T_m\}$	Set of Trainers, j = 1, ..., m
UIT_i	Unique identifier of Trainer T_i
$R = \{R_1, ..., R_i, ...R_k\}$	Set of Requesters, k = 1, ..., p
UIR_i	Unique identifier of Requester R_i
JR	A job requirement, defined in Definition 1

JC	A Job Contract
$OFcid_i$	An offer, defined in Definition 2
$UIOFcid_i$	Unique identifier for the offer $OFcid_i$
$UIJC_i$	Unique identifier for the job contract JC_i
$CIDM$	The content identifier of model M
$CIDT_iDP$	The content identifier of the Trainer T_i preview dataset DP
$MUcid_i$	The content identifier a model update

Definition 1 (job requirement)**.** JR: is the summary information of a job requirement, with the format of:

$JR = (Description, MaxValueByUpdate, MinRating, MinEvaluations, Tags,$
$NumberOfCandidates)$,

where $Description$ is the content describing the job information, $MaxValueByUpdate$ is the max acceptable value to pay for each model update, $MinRating$ is the minimum rating a Trainer must have to be considered as a candidate, $MinEvaluations$ is the minimum number of evaluations that a Trainer must have to be considered as a candidate, $Tags$ is a list of tags that will be used to calculate the scores of each Trainer, and $NumberOfCandidates$ is the number of candidates to be returned for the job.

Definition 2 (offer)**.** $OFcid_i$: is the necessary information to make an offer to a Trainer, the format of an offer is:

$OFcid_i = (Description, CIDM, ValueByUpdate, NumberOfUpdates)$,

where $Description$ is the content describing the offer information, $CIDM$ is the content identifier of the model, $ValueByUpdate$ is the max acceptable value to pay for each model update, and $NumberOfUpdates$ is the number of updates requested to the Trainer.

Definition 3 (summary trainer)**.** ST_i: is the summary information of the Trainer T_i:

$STi_i = (Description, DatasetSize, Processor, RAM, CPU, Tags,$
$MinValueByUpdate, CIDT_iDP, \&T_i)$,

where $Description$ is the content describing the own Trainer dataset, $DatasetSize$ is the dataset size, $Processor$, RAM and CPU are the Trainer PC specifications, $Tags$ is a list of tags that helps to describe and categorize the Trainer dataset, $MinValueByUpdate$ is the minimum acceptable value for each calculated update, and $CIDT_iDP$ is an **optional** content identifier of the Trainer preview dataset, a small public dataset accessible for users inside the organization. Finally, $\&T_i$ is the address of the Trainer, and it is filled by DAO.

Definition 4 (Job Contract)**.** $JCcid_i$: is the information available in a Job Contract JC, the format of a Job Contract is:

$$JCcid_i = (\&T_i, \&R_i, Description, CIDM, ValueByUpdate, NumberOfUpdates, NumberOfUpdatesDone, LockedAmount, AvailableAmount, Status),$$

where $\&T_i$, is the address of the Trainer, $\&R_i$ is the address of the Requester, $Description$ is the content describing the job contract, $CIDM$ is the content identifier of the global model, $ValueByUpdate$ is the agreed value to be paid for each model update sent by Trainer, $NumberOfUpdates$ is the agreed number of updates, $LockedAmount$ is how much is locked in DAO for this contract (on the begin it is always $ValueByUpdate * NumberOfUpdates$), $AvailableAmount$ is the amount that the Trainer can withdraw, and $Status$ is the current status of the Job Contract.

4.2 Entities

This section introduces the entities that compose the entire system and also describes their responsibilities and actions within the system. The section starts describing the two actors of the system Requesters and Trainers, in Sub Sections 4.2.2 and 4.2.1, respectively, and finally, Sub Section 4.2.3 describes DAO, the entity responsible for operating the system. The entities' implementations are available on the project repository: https://github.com/luccardozo/PlataformFL/.

4.2.1 Requesters

Any individuals or organizations that wish to train a model by federated learning. A Requester can match the candidates inside the organization that best fit a specific job, and then select the winners and submit an offer requesting the execution of a machine learning task, where the Requester pays once the task is performed. A Requester, once registered, can perform actions (function calls) within the organization, these actions are detailed in Table 4.2.

Table 4.2 – Requester Actions

Action	Function
$MatchTrainers(JR)$	Match the best Trainers for the job JR
$MakeOffer(OFcid, UIT_i)$	Make a new offer $OFcid$ to Trainer UIT_i
$RemoveOffer(UIOFcid_i)$	Remove the pending offer $UIOFcid_i$
$SignJobContract(UIJC_i, Amount)$	Sign the contract $UIJC_i$ and lock the contract $Amount$
$GetJobContract(UIJC_i)$	Get the details of the Job Contract $UIJC_i$
$CancelJobContract(UIJC_i)$	Cancel the Job Contract $UIJC_i$
$UpdateGlobalModel(UIJC_i, CIDM)$	Update the global model of $UIJC_i$ with $CIDM$

4.2.2 Trainers

Individuals who sell their computing power and/or their dataset to execute machine learning models for a Requester. A Trainer, once accepting a job offer, downloads a Requester's model and runs it on its own data, generating local updates and sending them to the Requester in exchange for payments. A Trainer, once registered, can perform the actions detailed in Table 4.3 in DAO.

Table 4.3 – Trainers Actions

Action	Function
$GetPendingOffers()$	Get all the pending offers
$AcceptOffer(UIOFcid_i)$	Accept the offer $UIOFcid_i$
$DeclineOffer(UIOFcid_i)$	Decline the offer $UIOFcid_i$
$SignJobContract(UIJC_i)$	Sign the contract $UIJC_i$
$SendUpdate(MUcid_i, UIJC_i)$	Send the update $MUcid_i$ to the job contract $UIJC_i$
$GetJobContract(UIJC_i)$	Get the details of the Job Contract $UIJC_i$
$CancelJobContract(UIJC_i)$	Cancel the Job Contract $UIJC_i$

4.2.3 DAO

The most important smart contract in the system, deployed over Layer 2, whose responsibility is to operate the entire system, this means that every interaction is done by

function calls inside this smart contract. The DAO responsibility is to perform Trainers payments and to lock the amount of a Job Contract, paid by Requester. It not only works as a middle-ware between Requester and Trainers communications, but it is also the only entity that can modify the status of the blockchain.

Apart from that, it is DAO responsibility to find the best Trainers that fit a specific job requirement (JR) of a Requester. It is done through a $MatchTrainers(JR)$[1] function call, and its implementation is shown in Algorithm 1. There is evaluated the scores of each Trainer that has 1) $Rating$ greater than or equal to the minimum rating required in $JR.MinRating$; 2)Has the minimum value accepted, $MinValueByUpdate$, less than or equal to the $MaxValueByUpdate$ of the Job Requirement; 3) Has at least $JR.MinEvaluation$, and returned only the $JR.NumberOfCandidates$ best candidates.

Algorithm 1 $MatchTrainers(JR)$, match the best Trainers that fit the requirements JR

Input: Job Requirement *JR*, define in Definition 1.
Output: `candidatesArray`, an array of ST_i, define in Definition 3, with the *JR.NumberOfCandidates* that best fits the requirements *JR*.

```
trainersScore is a list of Trainers.Length tuples (Trainer, integer).
for T in Trainers do
    if T.MinValueByUpdate ≥ JR.MaxValueByUpdate then
        continue
    end if
    if T.Rating < JR.MinRating then
        continue
    end if
    if Length(T.Evaluations) < JR.MinEvaluations then
        continue
    end if
    score ← CalculateScore(T, JR)          ▷ Achieve all requires, so calculate the score
    trainersScore.insert(T, score)
end for
trainersScore.sortAscByScore()             ▷ Sort asceding the array by scores
for i in JR.NumberOfCandidates do          ▷ Get just the best JR.NumberOfCandidates ST_i
    trainer ← trainersScore[i][0]
    candidatesArray[i] = trainer.getSummary()
end for
return candidatesArray
```

Algorithm 2 shows the implementation of $CalculateScore(T, JR)$, which is used inside Alg. 1 to calculate the score of each Trainer (T) for a specific JR. The score of a Trainer is a float number between 0 and 1, and is composed of *tagScore*, *ratingScore* and *priceScore*, as follows:

[1] https://github.com/luccardozo/PlataformFL/

- **40% tagScore:** Represented the proportion of matched tags, Trainer if more matched tags will have a better $tagScore$.
- **30% ratingScore:** Represented the Trainer rating, Trainer that has a better rating will have a better $ratingScore$.S
- **30% priceScore:** Represented the spread between the $JR.MaxValueByUpdate$ and the Trainer $T.MinValueByUpdate$. Trainer lower minimum price will get a higher spread and a better $priceScore$.

Algorithm 2 $CalculateScore(T, JR)$, Calculate the score of trainer T for the requirement JR

Input: Trainer *T* and Job Requirement *JR*, define in Definition 1.
Output: `score`, the calculated Trainer score 0 up to 1.

$matchedTags \leftarrow 0$
for `tag in Trainers.Tags` **do**
  if `tag in JR.Tags` **then**
    $matchedTags \leftarrow matchedTags + 1$
  else
    continue
  end if
end for
$tagsScore \leftarrow \frac{matchedTags}{Length(JR.Tags)} \times 0.4$
$ratingScore \leftarrow \frac{T.Rating}{5} \times 0.3$
$priceScore \leftarrow \frac{JR.MaxValueByUpdate - T.MinValueByUpdate}{JR.MaxValueByUpdate} \times 0.3$
$score \leftarrow tagsScore + ratingScore + priceScore$
return $score$

4.3 Blockchain Structure

This section first introduces the selected networks that compose the blockchain structure, then explains in detail the reasons behind the choices, and finally, describes how the two networks work together, pointing out the strengths of the choice.

The blockchain structure is composed of Layer 2 Optimistic Roll-up running on top of a public blockchain. The choice of a public blockchain was made because, when compared with a private blockchain, the public blockchain has a higher degree of decentralization and privacy. It also operates on an incentivizing scheme that encourages new participants to join and keeps the network agile. The choice of Optimistic Rollup as Layer 2 was made because the platform concerns are about decentralization, security, privacy, cost per transaction, scalability, and coding difficulty (see Section 2.4). With this relevance, Optimistic-Rollup leads the race. Even though ZK-Rollup could be a good choice it is still a very new technology that has coding difficulty too high. None of the known

solutions have full integration with Solidity, the language used to write smart contracts on Ethereum, yet.

The public blockchain chosen as Mainnet (the bottom layer) was the Ethereum[2] network and for Layer 2 the selected one was Arbitrum[3], which is an Optimistic Roll-up solution. Figure 4.2 shows an overview of the blockchain structure used in this work; here both blockchains are represented by a blue cloud, and the rollup process from Layer 2 to the bottom layer is represented inside a square between the blockchains. The DAO smart contract is deployed inside Layer 2, and all the transactions are done through it.

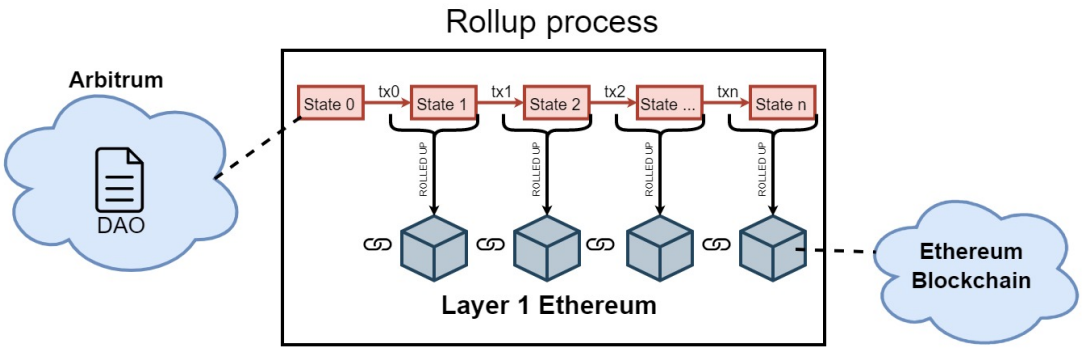

Figure 4.2 – Blockchain structure

Ethereum is a good choice not only because its digital currency, Ether, is accepted by most people but also because it is the one with a more long-term value, a more mature environment, more development tools, and more credibility than the others. However, not everything is good and the drawback of Ethereum is that as the number of transactions increases, the gas fee for each transaction also increases. This drawback is addressed by the use of Arbitrum Layer 2, which provides a complete Ethereum-compatible chain, a much lower gas fee, and a higher TPS while relying on the security of the bottom layer.

By the use of the major public blockchain, with the biggest active community and with its own cryptocurrency, this blockchain structure aims to make this platform not only accessible for everyone but also a secure environment with long-term value for every user that wants to train a model or sell their computation power to realize a machine learning task. On the other hand, using Arbitrum Layer 2 the blockchain structure aim to reduce the gas fee spent over the transaction making the platform cheaper and more scalable.

[2]https://ethereum.org/en/
[3]https://bridge.arbitrum.io/

4.4 Transaction Process

This session will explain how the system works. For this, the use case of Figure 4.3 is used. There, the Requester first registers in the system and then requests the service of training a global model to a Trainer. The Trainer first registers on the system and then accepts the offer. To make the explanation easier, each step of the process shown is numbered with a value.

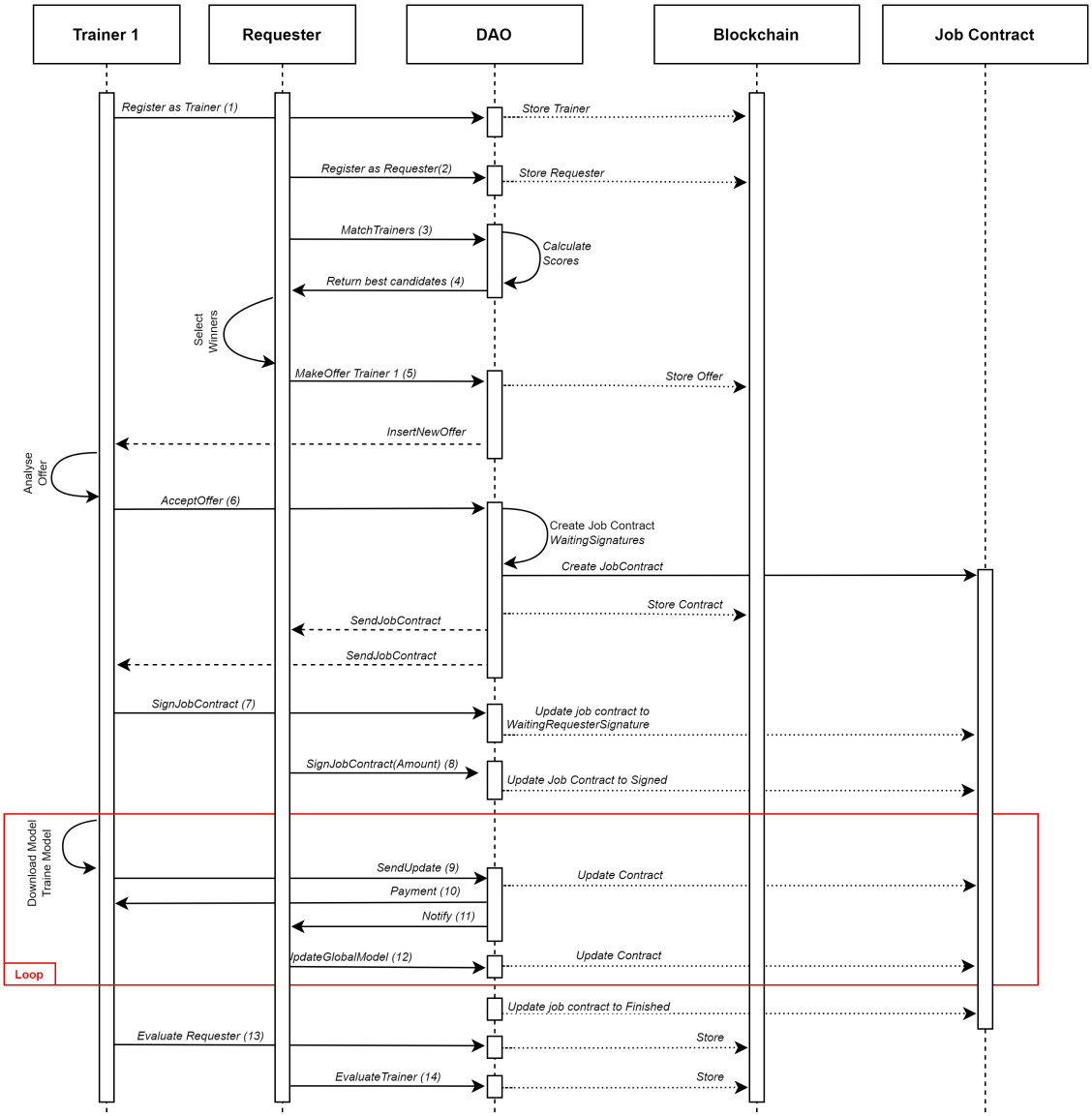

Figure 4.3 – Sequence diagram of the transaction process

In the first step (1), a Trainer is registered on the platform by interacting with the DAO by calling the $RegisterAsTrainer(ST_i)$, where ST_i is the Summary of the Trainer's Information and is defined in Definition 3. Once this is done, the Trainer and his ST_i are registered on the platform, and the DAO takes care of storing it on the blockchain.

Then, in step (2), a Requester is registered on the platform by interacting with the DAO by calling the $RegisterAsRequester$, once this is done the Requester is registered on the platform and the DAO takes care of storing it on the blockchain.

In step (3), the Requester interacts with the DAO by calling $MatchTrainers(JR)$, where JR is the Job Requirement, defined in Def. 1, to get the Trainers that best suit the requirements JR. $MatchTrainers(JR)$, shown in Alg. 1, calculate the score of each Trainer that full-fill the basic requirements of JR and return the best candidates. After DAO calculates the score of each Trainer that satisfy the Job Requirement, it returns the ST_i of the bests $JR.NumberOfCandidates$ candidates, on step (4). After receiving the best Trainers for the job ST_i , the Requester can analyze the ST_i of each one of them, and then select the winners by making an offer to them.

In step (5) the Requester makes an offer to the Trainer by interacting with the DAO by calling $MakeOffer(OFcid_i, \&T_i)$, where $OFcid_i$ is the offer information, defined in Definition 2, and $\&T_i$ is the address of the Trainer, received as a result of step (3) together with ST_i of the Trainer. Once the offer is done, DAO places it into the Trainer pending offers list, notifies and registers it on the blockchain.

As soon as the Trainer receives notification of a new offer, he calls DAO an $getPendingOffers()$ and gets all the pending offers for analysis. Once analyzed, in step (6), it accepts the offer by calling $AcceptOffer(UIOFcid_i)$, where $UIOFcid_i$ is the unique identifier of the offer. The DAO creates a Job Contract (JC), defined in Definition 4, with the state ***WaitingSignatures*** and sends it to Trainer and Requester to assign.

When a Requester and Trainer receives a Job Contract (JC) with unique identifier $UIJC_i$, they can analyze it and accept or decline the Contract. In step (7) Trainer sign the contract calling $SignJobContrat(UIJC_i)$, and in step (8) the Requester sign the contract calling $SigJobContract(UIJC_i, Amount)$, where $Amount = ValueByUpdate * NumberOfUpdates$. DAO locks the amount and updates the Job Contract to signed. From that, the locked $Amount$ will be used to pay for each update sent by the Trainer. By locking the $Amount$ it is guaranteed that the Trainer will always be paid when sending an update, also this makes it possible for both parties to cancel a work contract in progress and not be harmed. When there is a cancellation of a contract, the Requester takes all the remaining locked value.

Once the Job Contract is signed, it is Trainer's responsibility to generate the local model update by downloading the model and running it over their own data set. As soon as the Trainer obtains the local model update, $MUcid_i$, he stores it in a decentralized way

using IPFS and gets its Content Identifier(see Section 2.5) $CIDM$. With the Content Identifier of the model $CIDM$ on hands, the Trainer, in step (9) sends the calculated update by calling $SendUpdate(CIDM, UIJC_i)$. Then, DAO updates the Job Contract $UIJC_i$ by 1) incrementing the number of updates sent on the contract; 2) making the payment for the update to the Trainer, in step (10), and, finally, notifying the Requester, in step (11).

As soon as the Requester receives the notification, he downloads the Trainer update and generates the new global model without accessing any data. The Requester stores it and gets the CID of the new global model ($CIDM$). In step (12), the Requester calls $UpdateGlobalModel(UIJC_i, CIDM)$ to update the new global model of the Job Contract $UIJC_i$ with $CIDM$. After that, the looping, shown in red, starts again and the Trainer must download the new global model, generate the new local model update, and sent it to the DAO until the submissions of all the agreed local model updates are made or until one of the parties cancels the contract.

When all updates are sent, the DAO immediately updates the Job Contract status to *Finished*, not accepting more updates nor making payments. Once the Contract is finished, the Requester and Trainer can send their evaluation of the Job by calling $EvaluatePeer(UIJC_i, Description, Rating)$,in steps (13) and (14), where $UIJC_i$ is the Job Contract identifier, $Description$ is a free text to make a comment on the peer, and $Rating$ is the rating given to the peer. Finally, DAO store these evaluations on the Blockchain so that are available for the next jobs.

5 EVALUATION & RESULTS

This Chapter has the general objective of describing how the proof of concept of the proposed platform was implemented in a distributed learning application. First, Section 5.1 introduced the used tools and software for the platform implementation. Then, in Section 5.2 is described the test environment prototyping over evaluation. Section 5.3 describes the use-case for the test and the test scenario running over the platform. In Section 5.4 is shown the test evaluation, followed by some estimates, and finally, in Section 5.5 is shown and discussed the achieved results.

5.1 Enabling Technologies

Solidity programming language was used in conjunction with IDE Remix to implement the smart contracts, their code is available on the platform repository [1]. The Hardhat development tool, together with the VSCode code editor and the MetaMask wallet, was used to deploy and interact with smart contracts on Arbitrum's test network, RinkArby. To store the global model and the generated updates in a decentralized way using IPFS was used Pinata. The complete implementation was carried out on a WSL2 utilizing Linux Ubuntu 18.0 LTS from within a Windows 10 machine, and to simulate other entities inside the organization was used virtual machines with also Ubuntu 18.0 LTS installed.

- **Solidity 0.8.12:** Programming language for smart contracts that run on the public Ethereum blockchain. Used for implementing smart contracts, available at: https://github.com/ethereum/solidity/releases;
- **Hardhat 2.11.2:** Development environment for the Ethereum network. Used to deploy contracts and interact with the contract using the javascript programming language, available at: https://www.npmjs.com/package/hardhat/v/2.11.2;
- **Python 3.10.X:** Programming language used to implement and run the Convolution Matrix filter algorithm, available at: https://www.python.org/downloads/release/python-3100;
- **NumPy 1.22.3:** Offers comprehensive mathematical functions, random number generators, and linear algebra routines. Used to implement the Convolution Matrix

[1] https://github.com/luccardozo/PlataformFL

filter algorithm, available at: https://pypi.org/project/numpy/1.21.1/

- **VSCode 1.71.X:** Open source editing tool, available at: https://code.visualstudio.com/updates/v1_71
- **RinkArby:** Arbitrum's testnet that uses the same source code, but runs on top of Ethreum's Rinkeby testnet.
- **Rinkeby:** Ethereum test network that uses the same source code but uses a faceut process to distribute funds to wallets.
- **Pinata:** A plataform to save data on an IPFS node. Used to store the Requester global model or the model update generated by Trainer, available at: https://www.pinata.cloud/
- **MetaMask:** A cryptocurrency blockchain wallet. Used to create private chain accounts, available at: https://metamask.io/
- **Arbiscan:** An explorer to on-chain data for the Arbitrum testnet. It was used to monitor gas price, gas used, and the transaction fee of the network, available at: https://arbiscan.io/

5.2 Test Environment

The network used for the evaluation was the RinkArby network, which is an Arbitum test network that runs on top of the Rinkeby network, an Ethereum test network. Due to the need to spend real money for transactions, neither Ethereum nor Arbitrum production networks were used. However, the test networks run on the same code base as Mainnet and on these networks to get fake money (fake ETH in this case) it is only necessary to do a faceut[2] process. Once this process is done, the test network user earns funds and is able to make transactions/deployments on the network. Note that although the test network uses the same source code as the production network, the network delay of the test network is not the same as the production network.

The test environment designed for the validation of the platform implementation is shown in Figure 5.1 at a high level. It is composed of 5 peers connected on the same blockchain, 3 of which are Trainers, 1 Requester, and 1 Server. The Server is responsible for both deploying the DAO smart contract on the RinkArby network and to running the virtual machines. The DAO smart contract implementation is available

[2] A crypto faucet lets users earn small crypto rewards by completing simple tasks.

at: https://github.com/luccardozo/PlataformFL. The remaining peers are virtual machines connected on the same blockchain and that interact with DAO smart contract deployed by the Server.

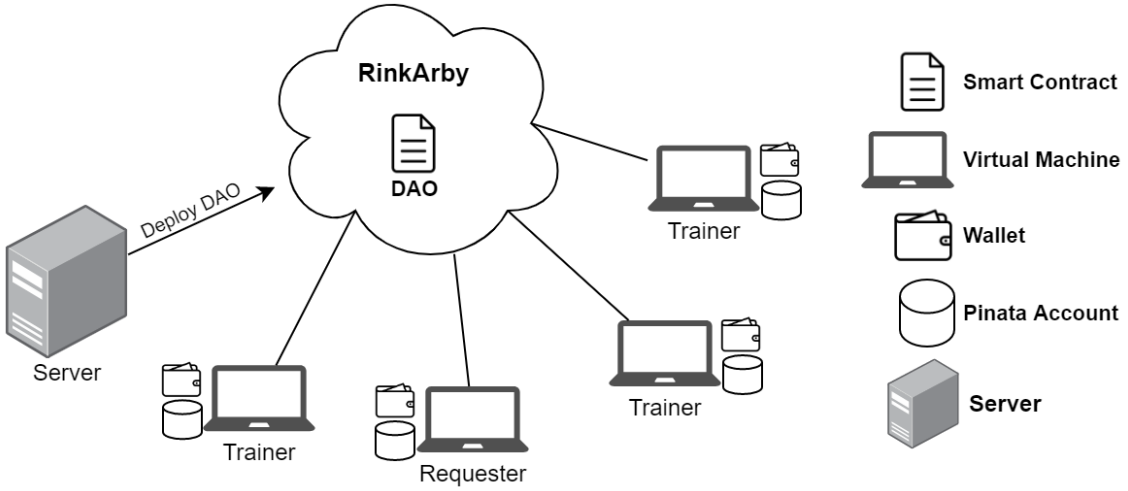

Figure 5.1 – Test Environment

For each virtual machine was created a private chain account, aka. "wallet", using MetaMask and they were all configured to connect on the same network, RinkArby - Arbitrum testnet. The used configuration on MetaMask is shown in Figure 5.2. A faucet process, using Rinkeby Faucet[3], was done for each VM in order to get funds into their wallet to transact.

Also, an account was created on Pinata for each virtual machine. These accounts are used to store the Requester global model(task) and the model update generated(task result) by the Trainer. Once the file is stored on Pinata, it automatically generates the file's CID, which is used to share the file with other peers.

To monitor the network and get statistics such as gas prices, gas used, and transaction fees, Arbiscan, an explorer for on-chain data on the Arbitrum testnet, was used. Figure 5.3 shows the transaction details of DAO smart contract deployment on the Arbitrum test network; the deployment of the DAO smart contract cost 3,424,656 units of gas, at a gas price of 0.0000000001 ETH (0.1 Gwei), for a total of 0.0003424656 ETH.

[3]A developer tool to get testnet Ether (ETH) in order to test and troubleshoot your decentralized application or protocol before going live on Ethereum Mainnet, where one must use real Ether. https://rinkebyfaucet.com/

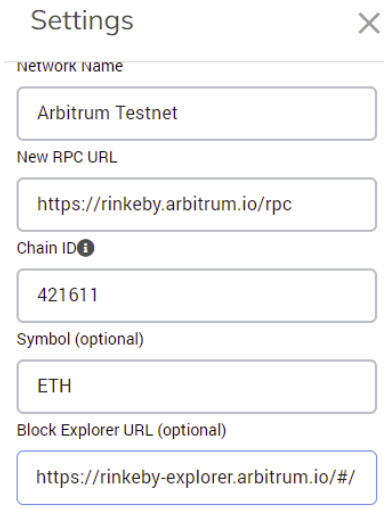

Figure 5.2 – MetaMask network configuration

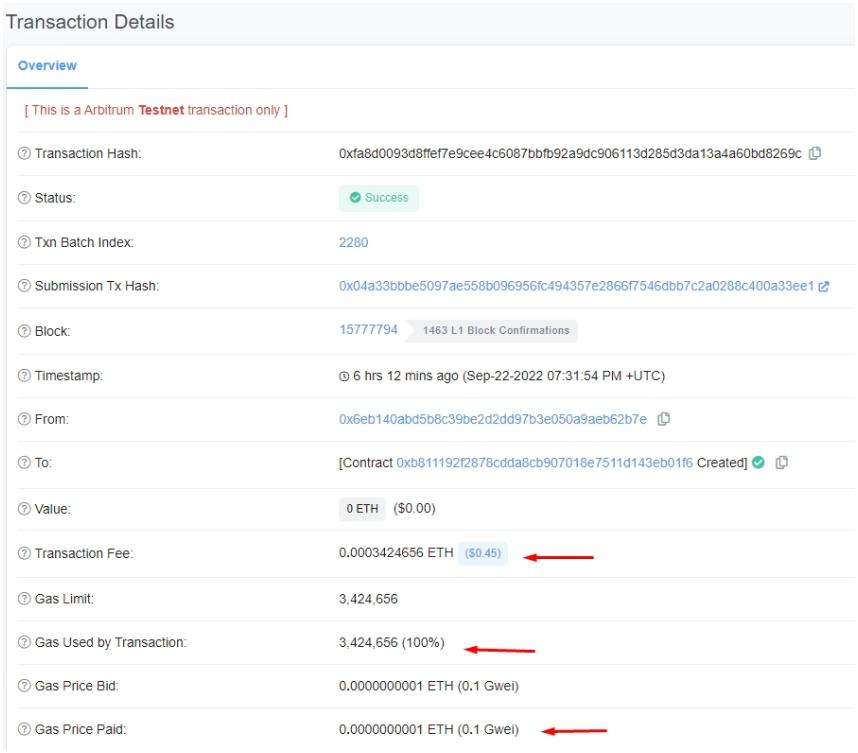

Figure 5.3 – Arbiscan: Transaction details of DAO smart contract deployment

5.3 Test Scenario

To achieve the objective, the proof of concept was done using a distributed Convolution Matrix filter, many used in Deep Learning. Figure 5.4 shows a convolution operation on a 4x4 matrix with a 3x3 kernel/filter using padding 1. In a convolution operation, the kernel first moves horizontally, then shifts down, and again moves horizontally. The sum of the dot product of the image pixel value and kernel pixel value gives the output matrix, on Figure 5.4 the first sum of the doc products is shown in red, the second one in

green, and the fifth in blue.

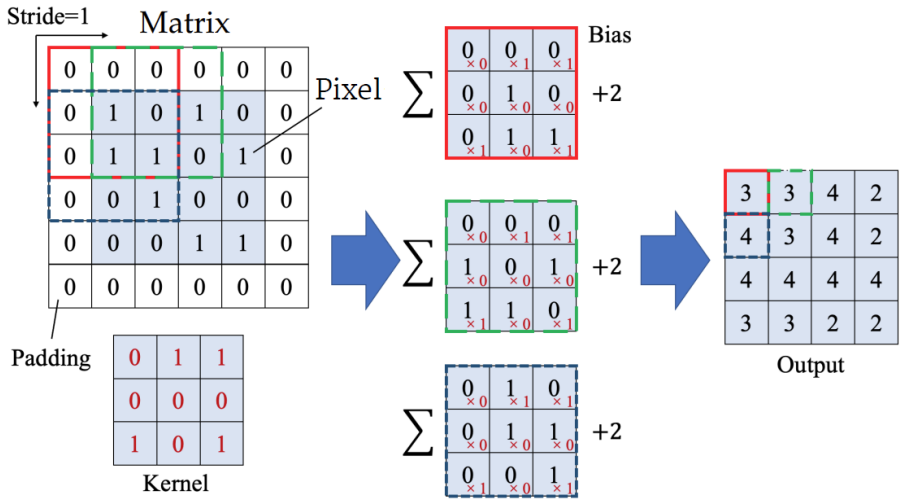

Figure 5.4 – Convolution operation of a 4x4 matrix with a 3x3 kernel. Source: (SHI et al., 2021)

As the value calculated for one pixel in the resulting matrix does not affect the value calculated for another pixel, the values of the resulting matrix can be calculated separately in a distributed way and then aggregated to generate the output matrix. This distributed process is the one used in this work to evaluate the platform.

In the executed test scenario, the Requester wants to run the convolution process on a 5x5 matrix with a 3x3 kernel/filter in a distributed way. The Requester requests service from 3 trainers in the organization, where each one is responsible for applying the kernel on just a few pixels and sending the result obtained to the Requester. At the end of the process, the requester aggregates all received updates to get the result of the entire process. The sequence diagram of this entire process is the same as presented previously in Section 4.4 but expanded to 3 Trainers and where the model is a convolution task.

The test execution and monitoring environment are the same as described in Section 5.1. The entire process is split into 5 tasks, each task calculates the convolution result over a group of strides. In this context, each task is a global model ($MUcid_i$), and each task result is a model update generated by Trainer.

Figure 5.5 shows the used matrix, kernel/filter, and the arrangement of pixels that will be covered for each task by the colors. Two of the Trainers are requested to realize two tasks each, while the remaining Trainer is requested to realize just one. The task assignments for each Trainer and their CIDs[4](to check the implementation of the tasks) is summarized as follows:

- **Trainer 1:** Responsible for realize Tasks 1 and 4.

[4]To access data on Pinata just use https://gateway.pinata.cloud/ipfs/<CID> replacing <CID>

$MUcid_1$: QmbpKij4aqK3yMF9QFN76kycsTa1bHfYfFQ7qB8uuphqGG

$MUcid_4$: QmeDQF2SKNceDq3ZRRo9n7iVA2DQEgZ9AuUeeMXgwQf3J8

- **Trainer 2:** Responsible for realize Tasks 2 and 5.

  $MUcid_2$: QmYsVZ6E5axx4U8ndPZGo3R9K5JTfpekYPeCpTExbNAnp4

  $MUcid_5$: QmQwPaHDtSeWqbmVToy8q3aMrjzxuR2NewfL3daVtKrPtr

- **Trainer 3:** Responsible for realize Task 3.

  $MUcid_3$: QmaSPY4zBwSpA3RAHMFwgVEZZ43X9sPpdBvZUJSMZRYYeo

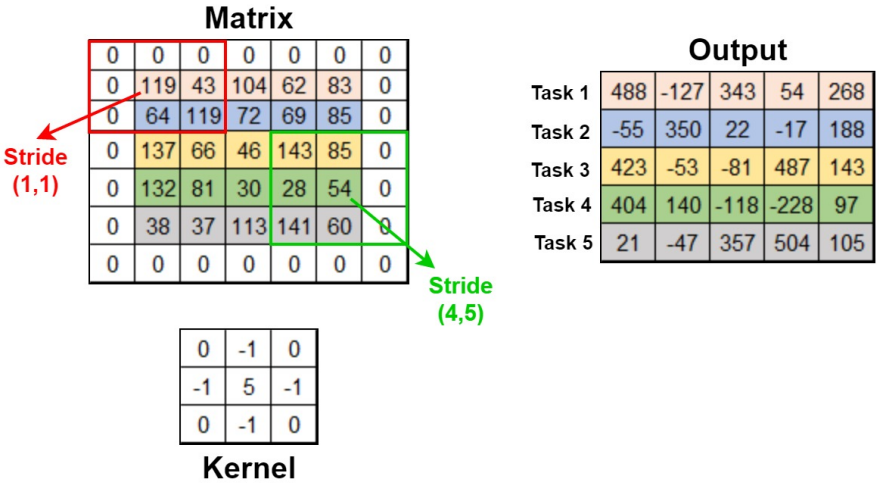

Figure 5.5 – Convolution operation test tasks

5.4 Evaluation

This section's goal is to evaluate the platform on the scenario introduced in the previous session, Section 5.3. To achieve this, this section first introduces the used metrics to calculate the transaction fee (gas fee). Then, it discusses the data captured from the interactions of peers within the organization during the Convolution Matrix filter process. Finally, it expands the achieved result to estimate how the number of Trainers affects the Requester gas fee expense.

Because each Ethereum transaction necessitates the use of computing resources, each transaction necessitates a fee. The charge necessary to complete an Ethereum transaction is referred to as gas. Gas fees are paid in ether, Ethereum's native currency (ETH).

- **Gas Used:** Refers to the amount of computational power required to execute a specified network operation.
- **Gas Price:** Is the price of a simple transaction, one gas.

- **Gas Fee:** Is the fee paid to execute a specific function on the blockchain, the same as transaction fee. Typically, $GasFee = GasUsed * GasPrice$

The gas cost and gas used for each function implemented on the proposed platform are shown in Table 5.1. The most expensive functions are those that modify the blockchain the most; functions that do not modify the blockchain, such as $MatchTrainer$ and $GetPendingOffers$, have no cost. The cost of registering a Trainer on the network is higher than that of registering a Requester since the Trainer must give his Summary Trainer (ST_I, see Def. 3) to be saved on the network.

Table 5.1 – Functions gas cost

Function	Gas used (unit)	Gas Fee (ETH)	Gas Price (ETH)
Deploy DAO	3424656	0,0003424656	0,0000000001
RegisterAsTrainer	1401532	0,0001401532	0,0000000001
SignJobContract	71544	0,000071544	0,0000000001
MakeOffer	67774	0,000067774	0,0000000001
UpdateGlobalModel	63047	0,0000630,47	0,0000000001
SendUpdate	51358	0,000051358	0,0000000001
EvaluatePeer	31580	0,000031580	0,0000000001
AcceptOffer	30895	0,000030895	0,0000000001
RegisterAsRequester	26937	0,000026937	0,0000000001
MatchTrainers	0	0	0,0000000001
GetPendingOffers	0	0	0,0000000001

Table 5.2 shows how many times each function was called by a peer, and Table 5.3 shows the total gas fee spent by a peer. The total gas fee spent over the Convolution Matrix process was 0,001673488 ETH. Both Trainer 1 and Trainer 2 spent 0,0003768882 ETH. It is because they have made the same number of tasks and their Summary Trainer used to register on the organization had the same size. On the other hand, Trainer 3 which made only one update spent less. The Requester is the one who spends the most on fees during the process, it is because he needs to make an offer and a contract for each Trainer, also he needs to send new tasks ($UpdateGlobalModel$) to Trainers 1 and 2. Observe that the number of $UpdateGlobalModel$ calls made is just 2, it is because the first 3 tasks are already attached in the contract, and just the other 2 tasks must be sent when a Trainer becomes free.

Another piece of information that we can take from Table 5.2 is that the minimum number of transactions happens when is requested for only one task to the Trainer, and it took 4 interactions: 1 $MatchTrainers$, 1 $MakeOffer$, 1 $SignJobContract$, and in the

end 1 $EvaluatePeer$. This means that the minimum number of interactions to make a deal with one Trainer is 3 interactions. The other interactions are all exchanges of updates.

Table 5.2 – Functions call

Function	**Requester**	**Trainer 1**	**Trainer 2**	**Trainer 3**
RegisterAsTrainer	0	1	1	1
AcceptOffer	0	1	1	1
SignJobContract	3	1	1	1
SendUpdate	0	2	2	1
EvaluatePeer	3	1	1	1
RegisterAsRequester	1	0	0	0
MatchTrainers	1	0	0	0
MakeOffer	3	0	0	0
UpdateGlobalModel	2	0	0	0
Total	13	6	6	5

Table 5.3 – Totals gas fee spent for each peer

	Requester	**Trainer 1**	**Trainer 2**	**Trainer 3**	**Sum**
Total Gas Fee (ETH)	0,000594181	0,0003768882	0,0003768882	0,0003255302	0,001673488

From Table 5.2 we can conclude that the Requester functions call to $MakeOffer$, $EvaluatePeer$, and $SignJobContract$ will always be equal to the number of Trainers he makes a deal with. This is obvious since to each Trainer it is necessary at least to make an offer, assign the contract and, in the end, send a peer evaluation. Also is known that the number of function calls to $UpdateGlobalModel$ will always be equal to the number of tasks minus the number of Job Contracts. This is obvious since one task is always attached to the contract, and the remains must be sent. That said, and using the gas fee calculated by each function call in Table 5.1 we can estimate how the number of trainers affects the gas fee expenses and the number of interactions for a requester.

Figure 5.6 estimates the gas fee spent and the number of interactions necessary to train a model with 50 tasks while variating the number of Trainers. When the number of Trainers is equal to 1, the number of interactions is equal to 52 (1 for making a trainer offer, 1 for signing the contract, 49 for sending the remains tasks, and 1 for the peer evaluation), otherwise, when the number of Trainers is equal to 50, the number of interactions go up to 150 (50 for make trainers offer, 50 for sign contracts, 50 for peer evaluation). It means that, the number of Requester interaction is equal to $(NumberOfTask - NumberOfTrainers) + (3 * NumberOfTrainers)$. We can also

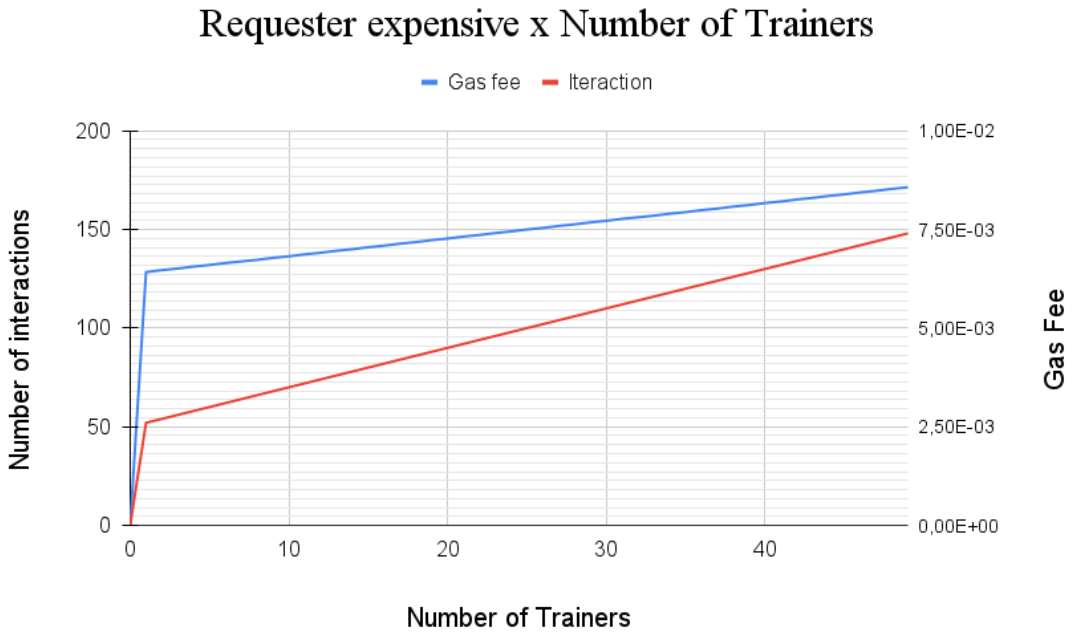

Figure 5.6 – Requester expensive x Number of Trainers for a model of 50 tasks

equate the estimated expenditure on transaction fees of each member of the network, it will be the sum of the number of all the functions called times the cost of each function ($\sum_{f=1}^{functions} funcionCalls_i * functionCost_i$). Below is a summary of the equations, there RI is the Requester interactions, TI is the Trainer interactions, and $GasFee_{member}$ is the estimated gas fee for a member.

$$GasFee_{member} = \sum_{f=1}^{functions} funcionCalls_i * functionCost_i$$

$$RI = (numberOfTasks - NumberOfTrainers) + (3 * NumberOfTrainers)$$

$$TI = numberOfTasks + 3$$

Also, Fig. 5.6 shows that increasing the number of Trainers affects less the number of interactions than the gas fee spent. Making it clear that making the process more distributed makes it more expensive. However, if more tasks are being done in parallel the process will probably be faster. In this work, the time that each function takes to be executed was not measured, and the subsequent estimation of data is based on the assumption that all transactions are executed at the same time interval.

If all transactions took the same amount of time to be executed and happened in perfect parallelism, where all Trainers took the same amount of time to conclude a task, we would arrive at Fig. 5.7 which shows that the number of rounds (a round is the

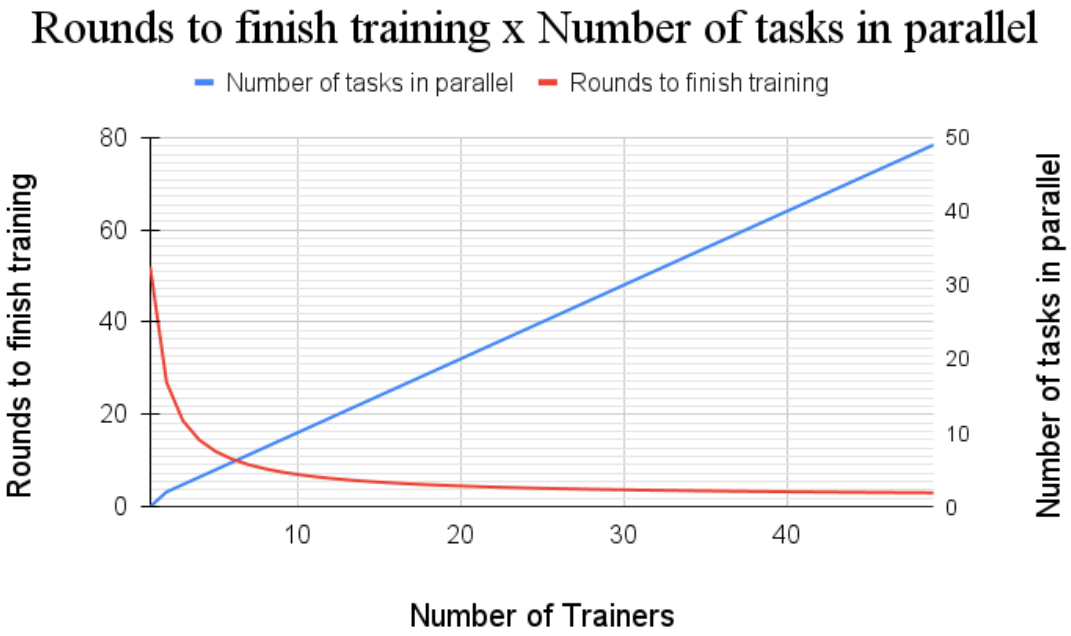

Figure 5.7 – Rounds to finish training x Number of tasks in parallel

conclusion of all tasks in parallel) drops as the number of tasks in parallel increases. With only 1 Trainer it will take 52 rounds, but with 5 Trainers the number drops to 5 rounds, because every round 5 tasks will be concluded, and with 50 Trainers it will take only 3 rounds (1 round to make trainers offer, 1 round to sign contract trainers contract, and 1 round to peer evaluation). Observe that the number of rounds decreases faster in the first Trainers that are added, and then the number of rounds starts to become more stable, making it less advantageous to continue increasing the number of Trainers. This is more clearly in Figure 5.8, where it shows how the number of tasks in parallel affect both the gas fee and the rounds to finish.

5.5 Results

The number of interactions to make a deal with a Trainer is only 3, and all the remaining interactions are exchanges of updates. In this way, the platform achieves a linear growth in the number of interactions as the number of Trainers grows. Another interesting point of the results, visualized in Figures 5.7 and 5.8, is that the platform scales better in relation to the gas fee spent, and rounds to finish a model training for the first Trainers that are added. This is because after a certain degree of decentralization adding a new Trainer does not increase the level of parallelism so much, but linearly affects the

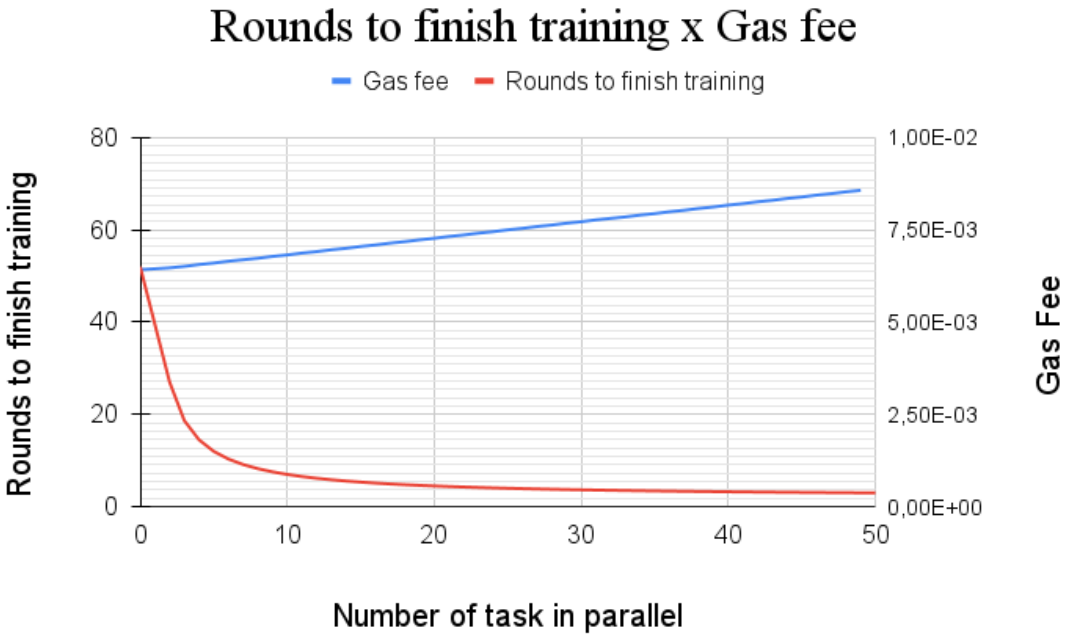

Figure 5.8 – Rounds to finish training x Gas fee

gas used making it less advantageous to add more Trainers.

When evaluating the proposed model front of (ZHENG et al., 2020) work, just the deployment of his platform had a cost of 0.0740 ETH, which is already more than 44 times greater than the total value of running the distributed convolution process on the proposed platform with 4 peers. Table 5.4 shows a comparison of the gas fee spent on the cheapest and most expensive functions of both solutions (note that, the (ZHENG et al., 2020) gas fee was adjusted to use the gas price of the day the tests were run, for a fair comparison). This is mainly for two reasons 1)(ZHENG et al., 2020) does not use any solution to improve the scalability and high-cost problem of the Ethereum network; 2) Its architecture uses an entity to intermediate the interactions, the addition of this entity increases the number of changes that must be recorded in the blockchain, making the gas used in his smart contracts bigger.

Table 5.4 – A comparison of the most expensive and cheapest functions of each model

function	Proposed Model	(ZHENG et al., 2020)
Most expensive	0,0003424656 ETH	0,0740 ETH
Cheapest	0,000026937 ETH	0,000684 ETH

The only other study studied that presents a similar solution is (FAN et al., 2021) work, however, he does not shows the spent on gas fee, nor the gas used in each smart contract, which makes direct comparisons of the results impossible. However, it is known

that his solution uses a payment channel to relieve the Ethereum network delay problem, it is also known that a payment channel to be opened requires the deployment of a smart contract on the bottom layer. This can be seen as a fragility from a cost perspective if the payment channel between the peers drops a lot, it will increase the number of deploys in the bottom layer, which has a high gas fee.

Table 5.5 compares the costs of the proposed platform functions running directly on the Ethereum test network with the values calculated in Table 5.1, running the platform on the Arbitrum Test Layer 2. As we can see the costs without the solution are 25 times higher.

Table 5.5 – Gas fee comparison between Ethereum and Arbitrum testnet

Function	Ethereum Testnet (ETH)	Arbitrum Testnet (ETH)
Deploy DAO	0,0085616	0,0003424656000
RegisterAsTrainer	0,0035038	0,0001401532000
SignJobContract	0,0000018	0,0000000715440
MakeOffer	0,0000017	0,0000000677740
UpdateGlobalModel	0,0000016	0,0000000630470
SendUpdate	0,0000013	0,0000000513580
EvaluateRequester	0,0000008	0,0000000315800
EvaluateTrainer	0,0000008	0,0000000315800
AcceptOffer	0,0000772	0,0000030895000
MatchTrainers	0	0
GetPendingOffers	0	0

6 CONCLUSION & FUTURE WORKS

This work aims to build a blockchain-based platform for federated learning, removing the need to centralize data on a server and encouraging edge nodes, called Trainers here, to sell their computing power to execute machine learning tasks. On this platform, Requesters can request for Trainers to perform machine learning tasks in exchange for cryptocurrency payments. The main requirements of this platform were to build something that has a low gas fee, autonomy, security, and privacy. Such requirements were met with the implementation of architecture with few entities, few message exchanges, and running on a Layer 2, which saves the transactions and then stores them in a public blockchain with greater security and decentralization.

The results obtained show that the platform is more economical in terms of gas rate than the others studied. This is mostly due to the simple communication process, where it only takes 3 interactions on the platform to finish contracting a Trainer and start training the model. This work is the result of research and the use of tools for the decentralization of information, which has its birth in the financial market area and great growth in the cryptocurrency and supply chain. Although some studies linking Blockchain to Federated Learning are being done, they are still few and most of them very theoretical in an environment of very fast growth. This work brings together some of the most recently studied technologies from the cryptocurrency environment and applies them to a platform for Federated Learning. The platform proposed Blockchain-based platform for Federated Learning was implemented and its feasibility was verified, the results obtained were analyzed and compared with similar models studied.

Although the initial proposal of a Blockchain-based platform for Federated Learning has been developed and verified, there are still many points that can be improved, added, and even analyzed. Thus, here are some suggestions for future work that could build on the work already done:

1. Integrate the codes developed with a high-level language, such as JavaScript and Python, to facilitate the execution of tests on a larger scale and to configure data capture to calculate latency and throughput. An example is a process of sending an update, which depends on the update being stored in a decentralized way using IPFS and generating CIDs. This process is very manual, and it can be automated using a high-level language with Solidity integration.
2. Apply the platform over an IoT environment, which is where the largest cluster of

nodes where federated learning takes place is in fact, and analyze results such as latency, scalability, and hardware necessities.

3. Implement a Voting system, giving the power to make changes within the organization to the members. While smart contracts were implemented in such a way as to facilitate the addition of a voting system feature, it was not implemented.

REFERENCES

AL xianxiongwang et. **LAYER 2 ROLLUPS**. 2021. [Online; last edit 28-October-2021]. Available from Internet: <https://ethereum.org/en/developers/docs/scaling/layer-2-rollups>.

BEATTIE, A. **A History of U.S. Monopolies**. 2021.

BITPAY. **What is the Lightning Network?** 2022. [Online]. Available from Internet: <https://bitpay.com/blog/what-is-the-lightning-network/>.

BUTERIN, V. **An Incomplete Guide to Rollups**. 2021. [Online]. Available from Internet: <https://vitalik.ca/general/2021/01/05/rollup.html>.

CHOY, D. **Project Strength: Which Blockchain Has The Highest Developer Count, And Why It Matters**. 2022. [Online; last edit 01-February-2021]. Available from Internet: <https://chaindebrief.com/which-blockchain-highest-developer-count/>.

COINMARKETCAP. **Criptocurency Market Cap**. 2022. [Online; last edit 22-September-2022]. Available from Internet: <https://coinmarketcap.com/>.

COINTELEGRAPH. **What is a decentralized autonomous organization, and how does a DAO work?** 2022. [Online]. Available from Internet: <https://cointelegraph.com/decentralized-automated-organizations-daos-guide-for-beginners/what-is-decentralized-autonomous-organization-and-how-does-a-dao-work>.

CSIRO. **Payment Channel (aka., State Channel)**. 2022. [Online]. Available from Internet: <https://research.csiro.au/blockchainpatterns/general-patterns/blockchain-payment-patterns/payment-channel/>.

DAI, W. et al. Sdte: A secure blockchain-based data trading ecosystem. **IEEE Transactions on Information Forensics and Security**, v. 15, p. 725–737, 2020.

DALY, L. **What Is Proof of Stake (PoS) in Crypto?** 2022. [Online; last edit 28-June-2022]. Available from Internet: <https://www.fool.com/investing/stock-market/market-sectors/financials/cryptocurrency-stocks/proof-of-stake/>.

DINH, T. T. A. et al. Untangling blockchain: A data processing view of blockchain systems. **IEEE Transactions on Knowledge and Data Engineering**, v. 30, n. 7, p. 1366–1385, 2018.

DOKU, R.; RAWAT, D. B.; LIU, C. Towards federated learning approach to determine data relevance in big data. In: **2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)**. [S.l.: s.n.], 2019. p. 184–192.

ETHEREUM. **Decentralized autonomous organizations**. 2022. [Online; last edit 01-Setember-2022]. Available from Internet: <https://ethereum.org/en/dao/>.

ETHEREUM. **The Merge**. 2022. [Online; last edit 15-Setember-2022]. Available from Internet: <https://ethereum.org/en/upgrades/merge/>.

EUROMONEY. **How does a transaction get into the blockchain?** 2020. [Online]. Available from Internet: <https://www.euromoney.com/learning/blockchain-explained/how-transactions-get-into-the-blockchain>.

FAN, S. et al. Hybrid blockchain-based resource trading system for federated learning in edge computing. **IEEE Internet of Things Journal**, v. 8, n. 4, p. 2252–2264, 2021.

HAN, R.; GRAMOLI, V.; XU, X. Evaluating blockchains for iot. In: **2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)**. [S.l.: s.n.], 2018. p. 1–5.

HOU, D. et al. A systematic literature review of blockchain-based federated learning: Architectures, applications and issues. In: **2021 2nd Information Communication Technologies Conference (ICTC)**. [S.l.: s.n.], 2021. p. 302–307.

HUA, G. et al. Blockchain-based federated learning for intelligent control in heavy haul railway. **IEEE Access**, v. 8, p. 176830–176839, 2020.

ISAAC, M.; FRENKEL, S. **Gone in Minutes, Out for Hours: Outage Shakes Facebook**. 2021. [Online; last edit 08-October-2022]. Available from Internet: <https://www.nytimes.com/2021/10/04/technology/facebook-down.html>.

JAKUB. **Rollups – The Ultimate Ethereum Scaling Solution**. 2021. [Online]. Available from Internet: <https://finematics.com/rollups-explained/>.

JUNG, T. et al. Accounttrade: Accountability against dishonest big data buyers and sellers. **IEEE Transactions on Information Forensics and Security**, v. 14, n. 1, p. 223–234, 2019.

KALODNER, H. et al. Arbitrum: Scalable, private smart contracts. In: **27th USENIX Security Symposium (USENIX Security 18)**. Baltimore, MD: USENIX Association, 2018. p. 1353–1370. ISBN 978-1-939133-04-5. Available from Internet: <https://www.usenix.org/conference/usenixsecurity18/presentation/kalodner>.

MA, C. et al. When federated learning meets blockchain: A new distributed learning paradigm. 09 2020.

MCMAHAN, B. **Federated Learning: Collaborative Machine Learning without Centralized Training Data**. 2017. [Online; last edit 06-April-2017]. Available from Internet: <https://ai.googleblog.com/2017/04/federated-learning-collaborative.html>.

MCMAHAN, H. B. et al. Federated learning of deep networks using model averaging. **CoRR**, abs/1602.05629, 2016. Available from Internet: <http://arxiv.org/abs/1602.05629>.

MONOLITH. **Understanding DeFi: Layer 2 explained**. 2021. [Online; posted 16-March-2021]. Available from Internet: <https://medium.com/monolith/understanding-defi-layer-2-explained-6981ef6c8990>.

MUSHARRAF, M. **What is InterPlanetary File System (IPFS)?** 2021. [Online; last edit 05-September-2022]. Available from Internet: <https://www.ledger.com/academy/what-is-ipfs>.

NAMBIAMPURATH, R. **Arbitrum vs. Optimism: What's the Difference Between These Ethereum Rollups?** 2022. [Online]. Available from Internet: <https://www.makeuseof.com/arbitrum-vs-optimism-whats-the-difference/>.

POWELL, J. S. . F. **Why Does Bitcoin Use So Much Energy?** 2022. [Online; last edit 18-Mayo-2022]. Available from Internet: <https://www.forbes.com/advisor/investing/cryptocurrency/bitcoins-energy-usage-explained/#:~:text=To\%20verify\%20transactions\%2C\%20Bitcoin\%20requires,intensive\%20than\%20many\%20people\%20realize.>

RABAH, K. V. O. Convergence of ai, iot, big data and blockchain: A review. In: . [S.l.: s.n.], 2018.

SALIMITARI, M.; CHATTERJEE, M. An overview of blockchain and consensus protocols for iot networks. **CoRR**, abs/1809.05613, 2018. Available from Internet: <http://arxiv.org/abs/1809.05613>.

SGUANCI, C.; SPATAFORA, R.; VERGANI, A. M. Layer 2 blockchain scaling: a survey. **CoRR**, abs/2107.10881, 2021. Available from Internet: <https://arxiv.org/abs/2107.10881>.

SHAYAN, M. et al. Biscotti: A ledger for private and secure peer-to-peer machine learning. **CoRR**, abs/1811.09904, 2018. Available from Internet: <http://arxiv.org/abs/1811.09904>.

SHI, J. et al. Improvement of damage segmentation based on pixel-level data balance using vgg-unet. **Applied Sciences**, v. 11, p. pp.518.1–17, 01 2021.

SHUTTLEWORTH, D. **What is a decentralized autonomous organization, and how does a DAO work?** 2021. [Online; last edit 07-October-2021]. Available from Internet: <https://consensys.net/blog/blockchain-explained/what-is-a-dao-and-how-do-they-work/>.

STAFF, C. **The Blockchain Trilemma: Fast, Secure, and Scalable Networks**. 2022. [Online; last edit 28-June-2022]. Available from Internet: <https://www.gemini.com/cryptopedia/blockchain-trilemma-decentralization-scalability-definition>.

STAFF, I. **What is IPFS?** 2021. [Online; last edit 22-July-2021]. Available from Internet: <https://docs.ipfs.io/concepts/what-is-ipfs/>.

VENTURES, O. **Blockchain Development Trends**. 2021. [Online; last edit 15-January-2021]. Available from Internet: <https://outlierventures.io/wp-content/uploads/2021/02/OV-Blockchain-Dev-Q1-2021-_v7.pdf>.

WANG, S. et al. When edge meets learning: Adaptive control for resource-constrained distributed machine learning. **CoRR**, abs/1804.05271, 2018. Available from Internet: <http://arxiv.org/abs/1804.05271>.

WIKIPEDIA. **Federated Learning**. 2022. [Online]. Available from Internet: <https://en.wikipedia.org/wiki/Federated_learning>.

XIAO, Y. et al. A survey of distributed consensus protocols for blockchain networks. **CoRR**, abs/1904.04098, 2019. Available from Internet: <http://arxiv.org/abs/1904.04098>.

YAGA, D. et al. **Blockchain Technology Overview**. [S.l.]: NIST Interagency/Internal Report (NISTIR), National Institute of Standards and Technology, Gaithersburg, MD, 2018.

YUE, L. et al. Big data model of security sharing based on blockchain. In: . [S.l.: s.n.], 2017. p. 117–121.

ZHENG, S. et al. A blockchain-based trading platform for big data. In: **IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)**. [S.l.: s.n.], 2020. p. 991–996.

ZOCHOWSKI, M. **Everything You Know about the Scalability Trilemma is Probably Wrong**. 2018. [Online]. Available from Internet: <https://medium.com/logos-network/everything-you-know-about-the-scalability-trilemma-is-probably-wrong-bc4f4b7a7ef>.

ZOCHOWSKI, M. **A Comparison Analysis of Layer2 Solutions**. 2021. [Online]. Available from Internet: <https://blog.zk.link/a-comparison-analysis-of-layer2-solutions-35dc5958253e?gi=ac65d47101d2>.