Flash loans in ethereum using Aave

Different use cases with specific analysis of applications

Advanced Studies in Finance / Sustainable Finance Master Thesis

Finance Executive Education

Finance Weiterbildung

Dr. Benjamin Wildiing

Manuel Keller

Written by:

Andrea Merli

Plattenstrasse 14

8302 Zürich

Filling date: XX.Januar.20YY

Universität Zürich^UZH

Abstract

abstract abstract. abstract .

Contents

1 Einleitung 1

1.1 Ausgangslage und Problemstellung . . . 1

1.2 Ziel der Arbeit . . . 1

1.3 Aufbau und Vorgehen . . . 1

1.4 Abgrenzungen . . . 1

2 Aave protocol 1

2.1 Aave lending protocol basics . . . 1

2.2 Aave Lending pool contract . . . 3

2.3 Flashloans . . . 4

3 Self hosted Ethereum node 5

3.1 Introduction . . . 5

3.2 Self hosted node vs provider . . . 6

3.3 Comparison of full nodes vs archive nodes in various ethereum implementations . 7

3.4 Motivation for choosing Erigon . . . 8

4 Blockchain explorers 10

4.1 Blockchain explorer usage . . . 10

4.2 Analyse tx in blockchain explorers . . . 11

4.3 Explorer selection . . . 11

4.4 Implikationen für Wissenschaft und Praxis . . . 11

5 LAST CHAPTER ENDING WITH NEW PAGE 12

5.1 Blockchain explorer usage . . . 12

5.2 Analyse tx in blockchain explorers . . . 12

5.3 Explorer selection . . . 13

5.4 Implikationen für Wissenschaft und Praxis . . . 13

6 Literaturverzeichnis 14

7 Anhang 15

List of Figures

1 P2P lending borrowing (OTC) 1
2 smart contract handling lender borrowed P2P interaction 1
3 Borrower repays the loan 2
4 Borrower fails to repay the loan 2
5 Aave standard lending borrowing 2
6 Monitoring in etherscan Aave Lending Pool contract 3
7 Example of a flashloan 4
8 Symplified veiw of ethereum nodes 6
9 Ethereum merge 9

List of Tables

1 Comparison Between ethLend and Traditional Finance 2
2 Comparison Between Aave V2 and Traditional Finance 3
3 LendingPool Aave V2 Contract Methods 4
4 Comparison Between Flashloans and Databases 5
5 Main characteristic Alchemy and Infura 7
6 Main characteristic Infura, Alchemy, and Self-Hosting a Node 7
7 Resource Requirements for Ethereum Clients 8
8 Erigon Advantages for Data Analysis 10

Abbreviations

1 Einleitung

1.1 Ausgangslage und Problemstellung

1.2 Ziel der Arbeit

1.3 Aufbau und Vorgehen

1.4 Abgrenzungen

2 Aave protocol

2.1 Aave lending protocol basics

Aave is one of the most used and battle tested lending protocols in DeFi. Before Aave version 1 (V1) there were other solutions for decentralised lending strategy, among them, ethlend which has been founded by the same team and it is the natural Aave precursor; ethlend was peer to peer (P2P), resulting in a direct lender - borrower matching handled by a smart contract. A traditional finance instrument to represent this is an OTC contract, represented in figure 1 to which the decentralized finance (DEFI) community added, at first, a layer for decentralisation and peers matching. The P2P approach enables the credit process in a decentralised environment but is not flexible and is illiquid cause the charged rate and lended amount in a potential loan must be published waiting for an interested counterpart.

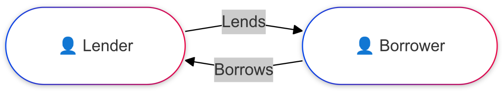


Figure 1: P2P lending borrowing (OTC)

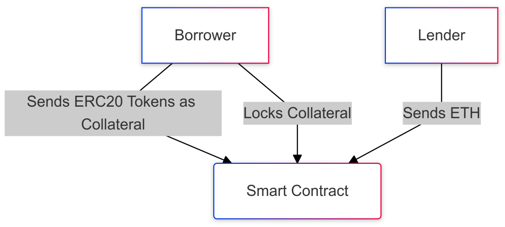


Figure 2: smart contract handling lender borrowed P2P interaction

In detail the OTC solution provided by ethLend, the aave precursor, represented in figure 2 consists in a lender borrower smart contract interaction where borrowers create a loan request posting ERC-20 compatible tokens as collateral and setting the loan's length, interest premium and the amount of tokens needed for collateral. If a lender agrees to these terms, a loan agreement will be created. There are only two possible outcomes: If the borrower repays the loan, the lender then receives his or her original principal plus interest and the borrower takes back the collateral which is unlocked from the smart contract whereas if the borrower fails to repay his or her loan, the lender will receive the borrower's posted collateral.

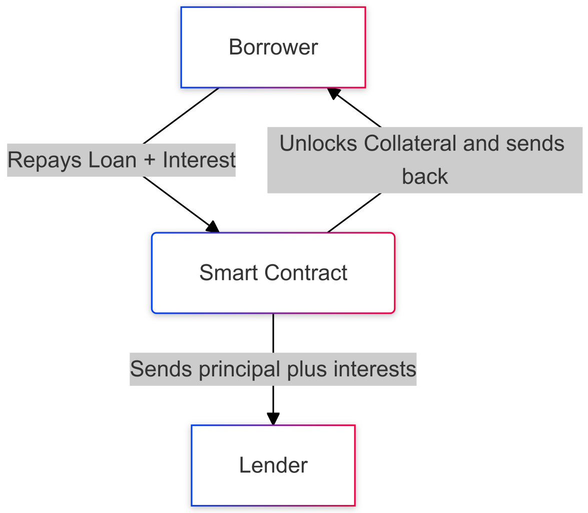

Borrower
Repays Loan + Interest
Unlocks Collateral and sends back
Smart Contract
Sends principal plus interests
Lender

Figure 3: Borrower repays the loan

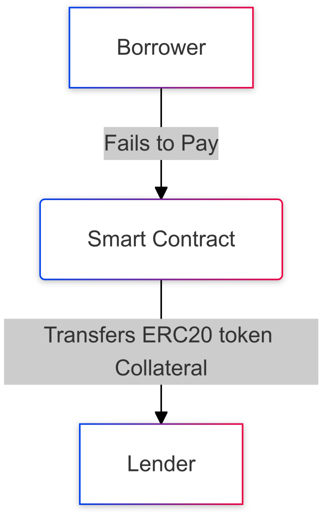


Figure 4: Borrower fails to repay the loan

Aave, from V1, breaks the P2P approach by creating lending pools where lenders deposit a cryptocurrency accepted by the protocol for the specific pool in exchange of a proxy token, receiving algorithmically calculated interest and borrowers borrow from the pool providing in exchange a collateral, chosen among the accepted ones from the protocol. Borrowing can be at variable or

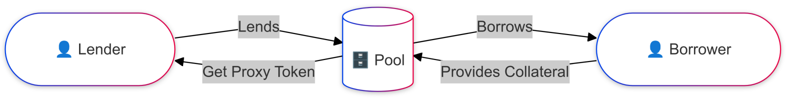


Figure 5: Aave standard lending borrowing

fix rate and each borrowing position has an health factor (H_f) which potentially triggers, based on an algorithm, a liquidation for an under collateralised loan (H_f below a calculated threshold). In traditional finance this can be similar to a margin call with the addition of an upgradable protocol driving the calls based on mathematical functions proposed through a governance process.

Table 1: Comparison Between ethLend and Traditional Finance

ethLend	**Traditional Finance**
Decentralized	Centralized
Peer-to-peer (P2P)	Over-the-counter (OTC)
Fail to pay: Liquidation	Fail to pay: Collateral swap

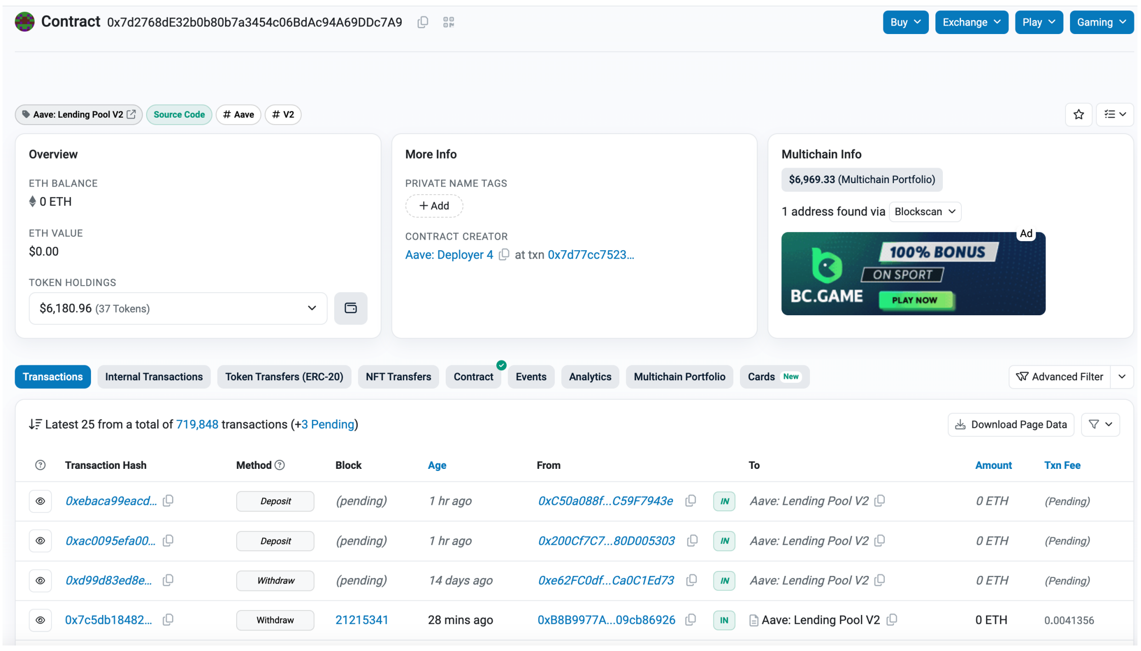
Aave V2 enhance V1 as gives the possibility of upgrading the proxy tokens given to lenders, reduces gas inefficiencies and simplifies code and architecture. V1 has the merit of flashloans introduction and V2 allows to use them for collateral trading, which means swapping collateral without closing and reopening a position, loan repayments, margin trading, debt swaps and margin deposits.

Table 2: Comparison Between Aave V2 and Traditional Finance

Aave V2	**Traditional Finance**
Decentralized	Centralized
Lending pool	(Derivative) market
Decentralized assessment of collateral	Collateral rating
Health Factor (H_f)	Margin definition
Collateral adjustments to avoid liquidation	Collateral adjustments to avoid liquidation
H_f close to 1: Alarm	Collateral <Margin: Margin call
Collateral increase or liquidation	Collateral increase or liquidation

This table compares Aave V2 and traditional finance approaches in lending and collateral management.

2.2 Aave Lending pool contract


Contract 0x7d2768dE32b0b80b7a3454c06BdAc94A69DDc7A9

Aave: Lending Pool V2 · Source Code · # Aave · # V2

Overview
ETH BALANCE
0 ETH
ETH VALUE
$0.00
TOKEN HOLDINGS
$6,180.96 (37 Tokens)

More Info
PRIVATE NAME TAGS
+ Add
CONTRACT CREATOR
Aave: Deployer 4 at txn 0x7d77cc7523...

Multichain Info
$6,969.33 (Multichain Portfolio)
1 address found via Blockscan

Latest 25 from a total of 719,848 transactions (+3 Pending)

Transaction Hash	Method	Block	Age	From		To	Amount	Txn Fee
0xebaca99eacd...	Deposit	(pending)	1 hr ago	0xC50a088f...C59F7943e	IN	Aave: Lending Pool V2	0 ETH	(Pending)
0xac0095efa00...	Deposit	(pending)	1 hr ago	0x200Cf7C7...80D005303	IN	Aave: Lending Pool V2	0 ETH	(Pending)
0xd99d83ed8e...	Withdraw	(pending)	14 days ago	0xe62FC0df...Ca0C1Ed73	IN	Aave: Lending Pool V2	0 ETH	(Pending)
0x7c5db18482...	Withdraw	21215341	28 mins ago	0xB8B9977A...09cb86926	IN	Aave: Lending Pool V2	0 ETH	0.0041356

Figure 6: Monitoring in etherscan Aave Lending Pool contract

The smart contract handling the lending pool including flashloan calls is deployed on ethereum and can be tracked. It is the smart contract LendingPool with contract address in ethereum main network: 0x7d2768dE32b0b80b7a3454c06BdAc94A69DDc7A9.

The Lending Pool includes calls which allow to deposit, redeem, borrow, repay, swap rate, liquidate, calling flash loans among others.

This paper focus is on calls to LendingPool,flashloan from ethereum inception to block number 20399999 correspondent to the date 27 July 2024.

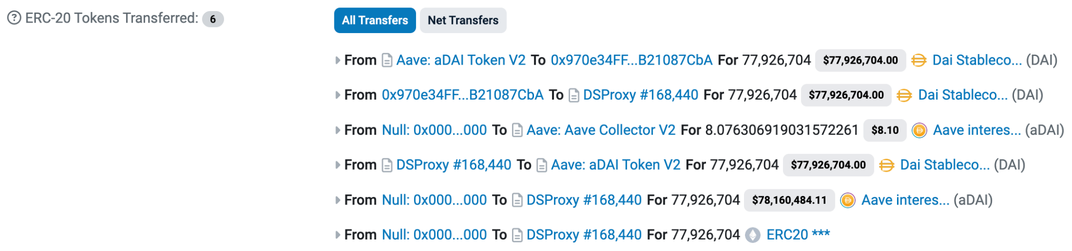
Table 3: LendingPool Aave V2 Contract Methods

Method Name	**Synthetic Description**	**Keccak Selector**
`deposit`	Supply assets to the LendingPool to earn interest.	`0xe8eda9df`
`withdraw`	Withdraw deposited assets from the pool.	`0x6b91c3e7`
`borrow`	Borrow assets from the pool against provided collateral.	`0x210dccae`
`repay`	Repay a borrowed asset to restore collateral usage.	`0xb7ea3af4`
`swapBorrowRateMode`	Switch between stable and variable interest rates.	`0x95db9357`
`liquidationCall`	Liquidate undercollateralized positions in the pool.	`0x3bde6c10`
`flashLoan`	Execute a flash loan with zero collateral requirements.	`0xee2e0890`

This table summarizes key methods of the Aave V2 LendingPool contract, including their Keccak-encoded selectors.

2.3 Flashloans

When a loan is issued a lending protocol as well as in traditional finance is expecting a collateral or a cashflow as a guarantee. Flashloans are unique in the blockchain environment as they don't require neither a collateral nor a cash flow, provided that the loan is paid, interests included, within the same ethereum transaction. If the loan is not repaid the transaction is rolled back.


ERC-20 Tokens Transferred: 6

From Aave: aDAI Token V2 To 0x970e34FF...B21087CbA For 77,926,704 $77,926,704.00 Dai Stableco... (DAI)
From 0x970e34FF...B21087CbA To DSProxy #168,440 For 77,926,704 $77,926,704.00 Dai Stableco... (DAI)
From Null: 0x000...000 To Aave: Aave Collector V2 For 8.076306919031572261 $8.10 Aave interes... (aDAI)
From DSProxy #168,440 To Aave: aDAI Token V2 For 77,926,704 $77,926,704.00 Dai Stableco... (DAI)
From Null: 0x000...000 To DSProxy #168,440 For 77,926,704 $78,160,484.11 Aave interes... (aDAI)
From Null: 0x000...000 To DSProxy #168,440 For 77,926,704 ERC20 ***

Figure 7: Example of a flashloan

The figure 7 shows the transactions wrapped in a flashloan one. It is possible to recognize the opening of the loan (first transaction) and the repayment with interests (third and fourth transactions). Flashloans are the main topic of this paper therefore is critical to clarify how they work. It is extremely common when reading documentation or articles to get confused by the statement that the repayment should happen within the same ethereum transaction: this can be misleading as with ethereum transaction, in this case is intended a transaction which can hold even hundreds of inner transactions as it is showed in Figure 7. This example shows a flashloan constituted of just six transactions as it has been chosen quite simple for the specific purpose. It is common at this point to be confronted with the question how can many transactions be performed and, in some cases, rolled back. To clarify this it is crucial to recall the definition of transactions and block. Transactions are intended to be cryptographically signed instructions sent from one Ethereum account to another where the account can be a public address associated to private keys typically controlled by a person or a contract account associated to a smart contract and controlled by code, whereas a block is a component of the chain, containing among the others a set of transactions and the information to rebuild the chain itself. The nature of blockchain permits the creation of flashloans cause if a set of operations, in this case inner transaction within the flashloan one, is not ending with the full repayment, all the operations will be aborted by not being added to the block, therefore not ending in the blockchain. The flashloan is therefore computed in a precommit phase and the persistency in the blockchain is subordinated to the repayment plus interests.. The described behaviour leads to the creation which doesn't exists in traditional finance and is comparable to a rollback in a traditional relational database where the atomic operation is the full bunch of inner transaction wrapped by the flash loan one.

Table 4: Comparison Between Flashloans and Databases

Flashloans	**Databases**
Decentralized	Centralized
Open flashloan, inner transactions, repay flashloans	Begin transaction, internal transactions, end transaction
Add or not to the block	Commit or rollback
Block added to the chain	Persist operation in DB

Flashloans have no counterpart in traditional finance. The best fit for a comparison is a transactional process in traditional databases.

Flashloans are not enabling the user to get an infinite loan. The constraint are the liquidity of the pool, as a bigger amount than the pool size itself cannot be accessed, and borrowing enabled for such pool.

3 Self hosted Ethereum node

3.1 Introduction

Ethereum is a distributed network of computers (known as nodes) running software that can verify blocks and transaction data. This is achieved by running on each computer of the network a client software which fullfills the ethereum specification. There are many implementation of such software and, provided that they fullfill all the protocol requirement, the nodes can interact independently from the implementation they run.

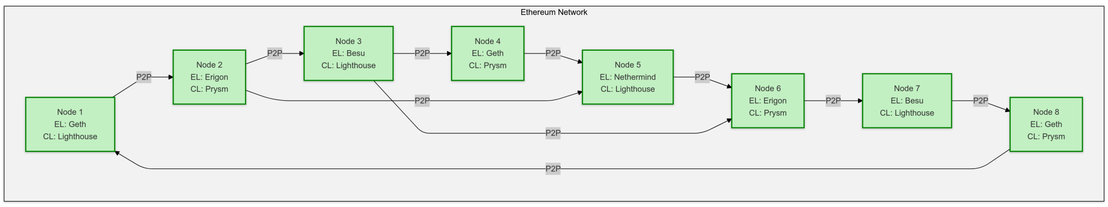


Figure 8: Symplified veiw of ethereum nodes

The figure 8 represents few nodes interacting. A node is any instance of Ethereum client software that is connected to other computers also running an Ethereum client software, forming a network. A client is an implementation of Ethereum that verifies data against the protocol rules and keeps the network secure. The figure 8 represents the current ethereum state with a transition from proof of work to proof of stake where a node has to run two clients, a consensus client and an execution client, which come with different possible implementations. The execution client, also known as the Execution Engine, EL client or formerly the Eth1 client, listens to new transactions broadcasted in the network, executes them in EVM (Ethereum Virtual Machine), and holds the latest state and database of all current Ethereum data. The consensus client, also known as the Beacon Node, CL client or formerly the Eth2 client, implements the proof-of-stake consensus algorithm, which enables the network to achieve agreement based on validated data from the execution client. All these clients work together to keep track of the head of the Ethereum chain and allow users to interact with the Ethereum network.

3.2 Self hosted node vs provider

This paper is written processing data from a self hosted node. That is not mandatory as it is possible to rely on services created ad hoc. Infura and Alchemy are popular third-party services that provide easy access to Ethereum nodes via APIs, allowing developers to interact with the Ethereum blockchain without running their own node. These services abstract the complexities of node management and provide scalable infrastructure for dApps, smart contract deployments, and blockchain analytics. Using such services has the advantage of providing a standardised environment out of the box but comes with some disadvantages. On a bulk analysis API calls can get expensive, there is not a full control on the data accessed, the providers can block access and there are centralisation risks which are exacerbated by relying fully on third party providers. Additionally externak providers implies a lack of control on data while a node implementation gives both the control and tools for accessing such data: being this paper based on data analysis, the choice of a local node provider suitable to self hosting has been prioritised. Running a node is not trivial at the current stage of the ethereum network. An archive node can require more than 10 TB disk space with certain implementations, a good bandwidth, fast disks and a computer with 32 Gb Ram and multicore. The next sections address the choice of node type and client implementation

Table 5: Main characteristic Alchemy and Infura

Alchemy	**Infura**
Advanced developer tools (e.g., debug API, enhanced APIs)	Basic Ethereum and IPFS API services
Performance optimizations (e.g., caching, load balancing)	Standard JSON-RPC API services
Customizable notifications and alerts	Basic webhook notifications
Higher-tier analytics for transaction debugging	Minimal analytics support
Enterprise-grade SLAs with premium support	Standard-tier SLAs

Table 6: Main characteristic Infura, Alchemy, and Self-Hosting a Node

Infura	**Alchemy**	**Self-Hosting a Node**
Cloud-hosted Ethereum and IPFS APIs	Enhanced APIs for analytics and debugging	Requires physical or virtual machine setup
No need for hardware or sync maintenance	Managed infrastructure with performance optimizations	Full control over node and data
Limited archive node access on basic plans	Advanced caching and query optimization	Can configure full or archive nodes as needed
Third-party dependency	Dependency on Alchemy services	Full decentralization and sovereignty
Quick and easy setup for development	Cloud-hosted, easy to scale	Longer setup time, requires technical expertise
Usage limited by rate limits and quotas	Developer-focused tools (e.g., Notify API, transaction explorer)	Unlimited usage, depends on hardware capacity

3.3 Comparison of full nodes vs archive nodes in various ethereum implementations

Ethereum nodes come in different types based on the amount of data they store and the roles they serve in the network. Running a self hosted node to perform data analysis requires to choose between a full node and an archive node.

A full node stores the complete current state of the Ethereum blockchain (i.e., account balances, contract storage, etc.) and recent historical data, but it prunes old state data to save space. Full nodes verify all transactions and blocks from the genesis block to the current state but don't keep every historical state like past balances or contract storage at every block. An Archive Node stores everything that a full node does, but in addition, it keeps all historical states for every block in the blockchain. This means archive nodes can provide the exact state of the blockchain at any point in history, but they require significantly more disk space. Full nodes are sufficient for most operations, while archive nodes are only necessary to access to the entire historical state of the blockchain.

A full node stores enough information to partecipate to the ethereum network, also as validator, but data analysis would become cumbersome as the stored information is enough to rebuild and validate the current state but requires computation to do it: for this paper the archive node is therefore the best choice.

3.4 Motivation for choosing Erigon

Table 7: Resource Requirements for Ethereum Clients

Client	Full Node Disk	Archive Node Disk	Full Node RAM	Archive Node RAM	CPU Requirements
Erigon	400–500 GB	~3.5 TB	2–8 GB	8–16 GB	Optimized (low-medium)
Geth	800–900 GB	~10 TB	4–16 GB	16 GB or more	High (sync heavy)
Besu	800–900 GB	~10 TB	8–16 GB	16–32 GB	Medium
Nethermind	700–900 GB	6–8 TB	4–8 GB	16 GB or more	Optimized (medium)
Lighthouse, Prysm	10–30 GB (Beacon)	N/A	1–4 GB (Beacon)	N/A	Low

This table lists the resource requirements for different Ethereum clients, including disk and RAM usage, and CPU demands for full and archive nodes.

There are many implementations of full archive nodes and the proper selection is related to data analysis suitability: save disk space, fast access to data, a client which allow to present the data in an human readable format for easy interpretation. For this purpose the most common implementations have been tested. The table 7 summarises resource requirements for the most common node implementations. Geth is the most widely used Ethereum client, Erigon is designed to be more resource-efficient than traditional Geth, especially in terms of disk usage and sync time. Besu is an Ethereum client written in Java, commonly used in enterprise environments. Nethermind is a high-performance Ethereum client written in C# with a focus on speed and configurability. Besu and Geth may require more memory depending on the workload.

For the present purpose, a combination of Geth and Lighthouse was tested (the latter only for consensus) as well as Geth and Besu; the test of geth and besu aborted while the second SSD disk installed as volume was half filled (max capacity 10 Tb and strong performance degradation).

Nevermind has not been tested as, being written in C#, is strongly suboptimal to work on a linux instance: the data related to it are the result of a direct query with chatgpt (the documentation of these tool is tends not to be aligned with the development and therefore to underestimate the resources required: LINK TO DOC AND OFFICIAL STORAGE). Considering the data presented in Table 7 Erigon looks the best fit, because it allows to run an archive node with a standard 4tb ssd disk. All ethereum node are very demanding in term of read write disk performance, therefore maximising the IO would be a plus, and two SSD in raid-0 are a good choice but not mandatory.

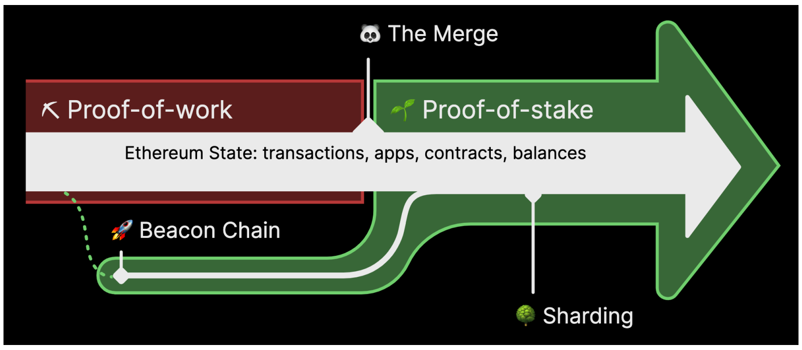


Figure 9: Ethereum merge

Another advantage of Erigon is having an embedded consensus node implementation. Since the merge, which is represented in figure 9, when ethereum became proof of stake, there is the need of an execution node and a consensus node. As showed in the picture for some time the consensus and execution layers run in parallel and they were consolidated in a unique chain with the merge; the consequence for data analysis is that the simplest is the consensus the best as the meaningful data are held in the execution layer. This is the case of caplin, the embedded consensus node in erigon, as it allows to achieve an archive node full synch without any configuration or installation of a compatible consensus node. Before starting the work in the main chain Erigon was tested on sepolia, a light test network, and verified that the data saved in a key value db, mdbx, were properly indexed and rendered by the embedded UI, otterscan. The last requirement for the analysis is being able to stop the synch at a certain block and run an rpc (remote procedure call) server which serves the data without synching continuously. Erigon satisfy also this. This allows to synch just once the main network till the desired block and then stop this operation, extremely resource demanding in term of bandwidth even when close to the ethereum chain tip. An additional section addresses otterscan compared to other ethereum UI like etherscan and blockscout. The table 8 summarises erigon advantage for self hosted data analysis:

Table 8: Erigon Advantages for Data Analysis

Advantage
Embedded consensus
Low disk usage
Fast read
UI integrated
Geth tools available out of the box

This table lists the key advantages of using Erigon for data analysis.

4 Blockchain explorers

4.1 Blockchain explorer usage

A blockchain explorer is a crucial tool for viewing, analyzing, and understanding data on a blockchain. It transforms complex, encoded information into human-readable form, making it easier to track transactions, verify addresses, explore smart contracts and can perform, in some case, some statistical analysis on behalf of the end user. Without a blockchain explorer, transaction data and smart contract interactions appear as hex-encoded information, which is not human-readable: below a set of pictures of a transaction in etherscan (the most used blockchain explorer), in otterscan (the explorer integrated in erigon) and raw data polling the node shows the difference between raw data and the same data rendered in an explorer the importance of utilising properly an explorer or a combination of them.

FIGURES HERE AND COMMENTS

The pictures show the same transaction: calling a Uniswap contract to swap tokens. An explorer decodes the data, showing details like token names, transfer amounts, and method names, so users can understand what actions are taking place. the raw data instead look like a series of encoded hexadecimal strings. Decoding is done programmatically using a set of rules known as the Ethereum Application Binary Interface (ABI). The ABI defines how to encode and decode function names and parameters for Ethereum smart contracts, allowing a blockchain explorer or other tools to convert raw transaction data into human-readable formats. Several libraries make ABI decoding relatively straightforward, especially for developers building tools or explorers: web3.js, ethers.js, Web3.py. For common contracts like Uniswap or ERC-20 tokens, the ABI files are usually public and can be added to the explorer's ABI database. These ABIs can be used to automatically identify functions and parameters, enabling accurate decoding.Blockchain explorers identify known smart contracts, such as Uniswap in the example above, by using their contract address and displaying relevant details, like contract name and functions. This allows users to recognise trusted contracts versus unknown ones, adding a layer of security when interacting with decentralized finance (DeFi) apps and other dApps. Explorers like Etherscan and Otterscan can show verified source code and contract metadata (if available) to give insights into the contract's purpose and functionality. In analysing a transaction sent to a Uniswap contract to perform a token swap the destination address (Uniswap's contract) and the hex-encoded transaction data (e.g., '0x18cbafe5' followed by more data) couldn't be interpreted easily. An explorer decodes this, revealing that '0x18cbafe5' maps to 'swapExactTokensForTokens' and decodes other parameters, showing token addresses, amounts, and recipient addresses.

In appendix xxx we show how to decode transactions in a purely programmatically way. This proofs the utility of explorers and shows what we nseed to do when explorer are not sufficient.

4.2 Analyse tx in blockchain explorers

Inhalt Inhalt

4.3 Explorer selection

4.4 Implikationen für Wissenschaft und Praxis

Inhalt Inhalt

Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt

5 LAST CHAPTER ENDING WITH NEW PAGE

5.1 Blockchain explorer usage

5.2 Analyse tx in blockchain explorers

Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt

Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt

5.3 Explorer selection

5.4 Implikationen für Wissenschaft und Praxis

Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt Inhalt
Inhalt Inhalt Inhalt Inhalt Inhalt

6 Literaturverzeichnis

Gantenbein, Pascal, und Marco Gehrig, 2007, Moderne Unternehmensbewertung, *Der Schweizer Treuhänder*, S. 602 – 612.

7 Anhang

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Mir ist bekannt, dass ich die volle Verantwortung für die Wissenschaftlichkeit des vorgelegten Textes selbst übernehme, auch wenn KI-Hilfsmittel eingesetzt und deklariert wurden. Alle Stellen, die wörtlich oder sinngemäss aus veröffentlichten oder nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Ort, den XX.XX.20YY

(Unterschrift der Verfasserin)