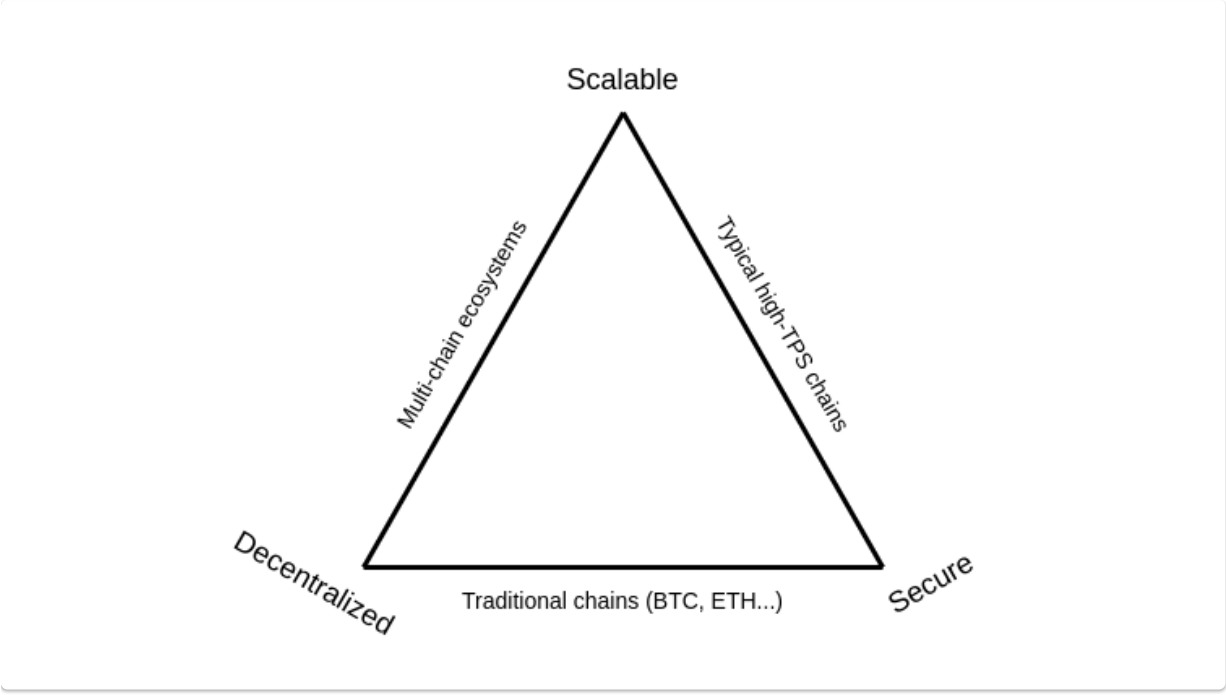
Lesson 7

Scalability Introduction

The scalability trilemma



"The decentralization of a system is determined by the ability of the weakest node in the network to verify the rules of the system." - Georgios Konstantopoulos

"For a blockchain to be decentralized, it's crucially important for regular users to be able to run a node, and to have a culture where running nodes is a common activity." - Vitalik article

On Ethereum there is a goal to keep the hardware requirements low.
There is a useful guide to scalability in their documentation

Approaches to Scalability

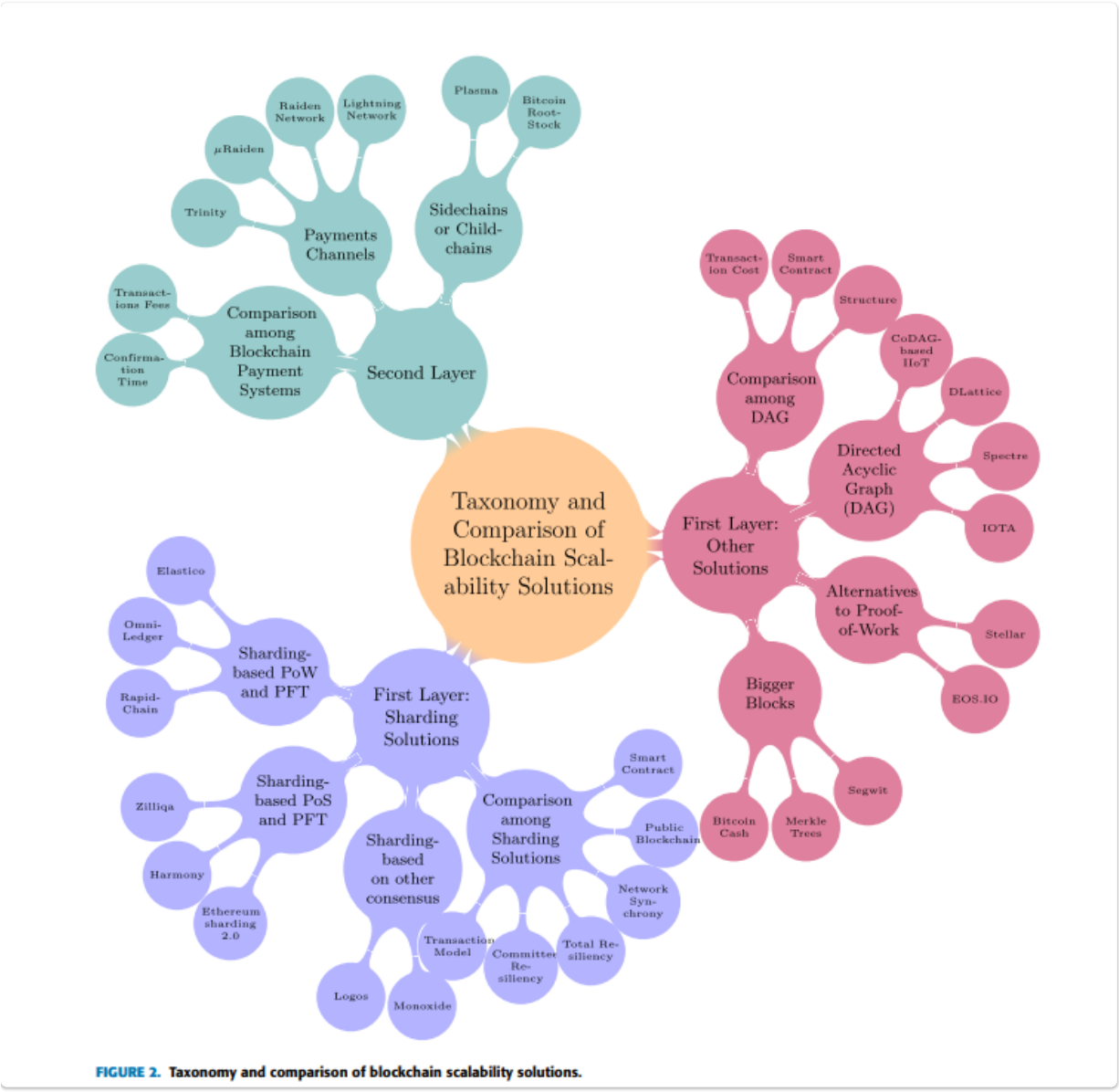


FIGURE 2. Taxonomy and comparison of blockchain scalability solutions.

From Scaling Blockchains: A Comprehensive Survey by Hafid et al.

Layer 1 Solutions

Tackling scalability at the L1 level is designing or redesigning your protocol to improve throughput, latency and finality.

Choice of Consensus Mechanism

Using a voting approach such as in BFT can have a negative impact on scalability. This is caused by the increase in the amount of messages being passed as the number of validators increases.
Therefore chains adopting a BFT based approach tend to use additional mechanisms to offset this.

On Ethereum validators form committees and votes are aggregated by committee.

Reducing transaction broadcasts

Solana moved away from using a gossip protocol to get transactions to the relevant parties. They reasoned that only the leader (block producer) needs to receive transactions, so on receiving transactions, nodes will send them to the leader rather than other nodes.

Parallel processing of transactions

In Ethereum transactions are ordered by the block producer and then executed sequentially.
This has the benefit of simplicity, but leads to

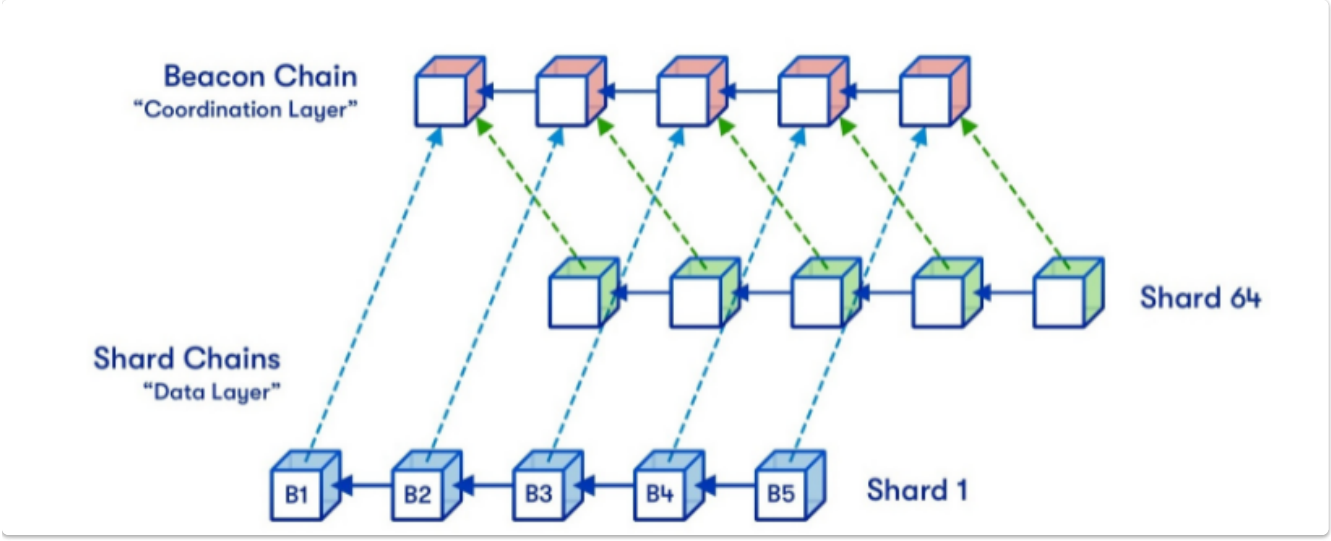
- MEV
- Poor horizontal scaling

More recent chains such as Solana / Aptos / Sui allow parallel processing of transactions.
They rely on an understanding of what areas of state will be

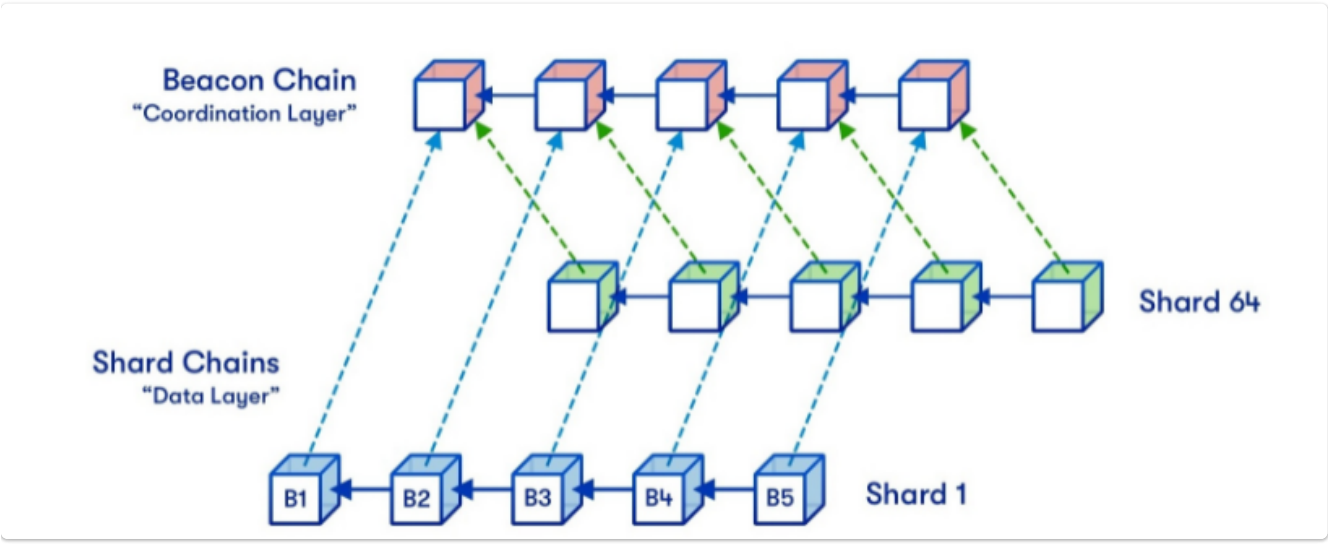
changed by a transaction, and therefore a dependency among transactions. From this they can allow 'non dependent' transactions to be processed in parallel.

Sharding

Ethereum plans to introduce 64 new shard chains, to spread the network load.



Vitalik's overview
Introduction



Introduction of Sharding

Vitalik sees 3 options

- Shards remain as data depots
- A subset of the 64 shards will allow smart contracts
- Wait until increased use of ZKPs allows private

transactions

Points from an article by Vitalik

There are three key limitations to a full node's ability to process a large number of transactions:

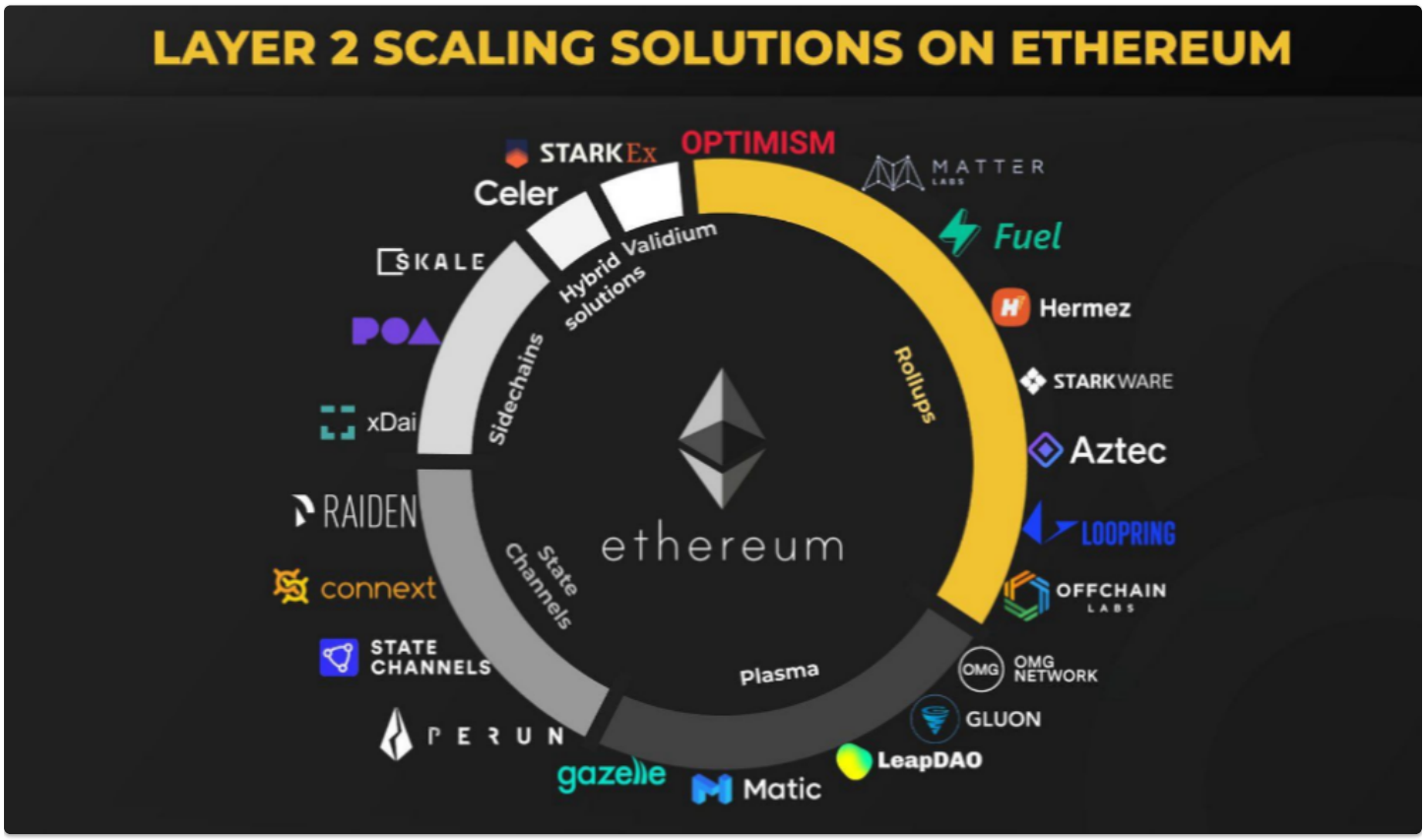
- **Computing power**: what % of the CPU can we safely demand to run a node?
- **Bandwidth**: given the realities of current internet connections, how many *bytes* can a block contain?
- **Storage**: how many gigabytes on disk can we require users to store? Also, how quickly must it be readable? (ie. is HDD okay or do we need SSD)

Many of the scalability approaches we see concentrate on the issue of computing power and how that can be made sufficient.

Off chain Scaling

Generally speaking, transactions are submitted to these layer 2 nodes instead of being submitted directly to layer 1 (Mainnet). For some solutions the layer 2 instance then batches them into groups before anchoring them to layer 1, after which they are secured by layer 1 and cannot be altered.
A specific layer 2 instance may be open and shared by many applications, or may be deployed by one project and dedicated to supporting only their application.

Early Layer 2 Solutions on Ethereum



Rollups

Rollups are solutions that have

- transaction execution outside layer 1
- data or proof of transactions is on layer 1
- a rollup smart contract in layer 1 that can enforce correct transaction execution on layer 2 by using the transaction data on layer 1

The main chain holds funds and commitments to the side chains
The side chain holds state and performs execution
There needs to be some proof, either a fraud proof (optimistic) or a validity proof (zk)

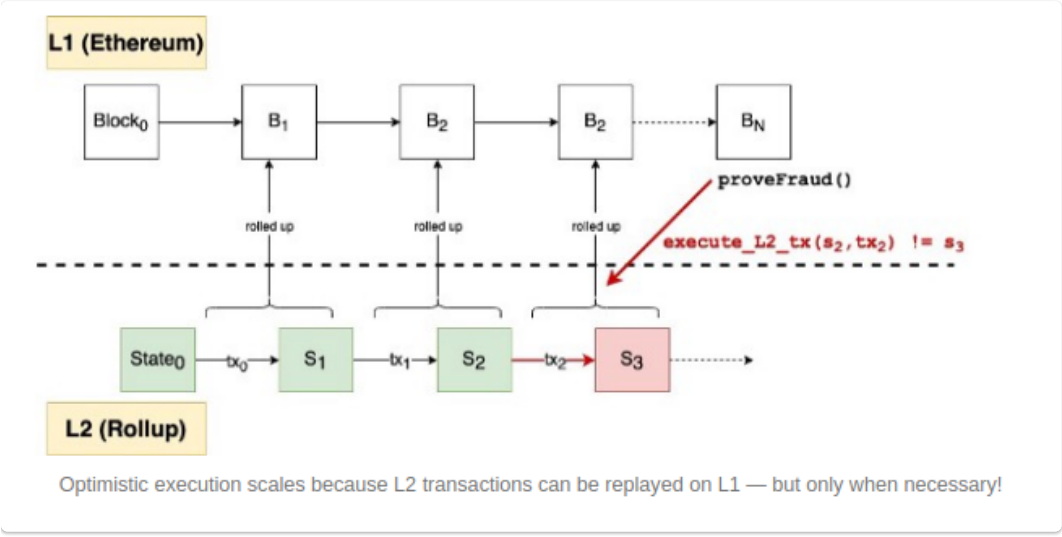
Rollups require "operators" to stake a bond in the rollup contract. This incentivises operators to verify and execute transactions correctly.

There are currently 2 types of rollups

- Zero Knowledge Proof rollups
- Optimistic rollups

Optimistic Rollups

The name Optimistic Rollups originates from how the solution works. 'Optimistic' is used because aggregators publish only the bare minimum information needed with no proofs, assuming the aggregators run without commiting frauds, and only providing proofs in case of fraud. 'Rollups' is used because transactions are commited to main chain in bundles (that is, they are rolled-up).



Optimistic execution scales because L2 transactions can be replayed on L1 — but only when necessary!

zkEVM Solutions

Rollup Recap

Rollups are solutions that have

- transaction execution outside layer 1
- transaction data and proof of transactions is on layer 1
- a rollup smart contract in layer 1 that can enforce correct transaction execution on layer 2 by using the transaction data on layer 1

The main chain holds funds and commitments to the side chains
The side chain holds additional state and performs execution
There needs to be some proof, either a fraud proof (Optimistic) or a validity proof (zk)

Rollups require “operators” to stake a bond in the rollup contract. This incentivises operators to verify and execute transactions correctly.

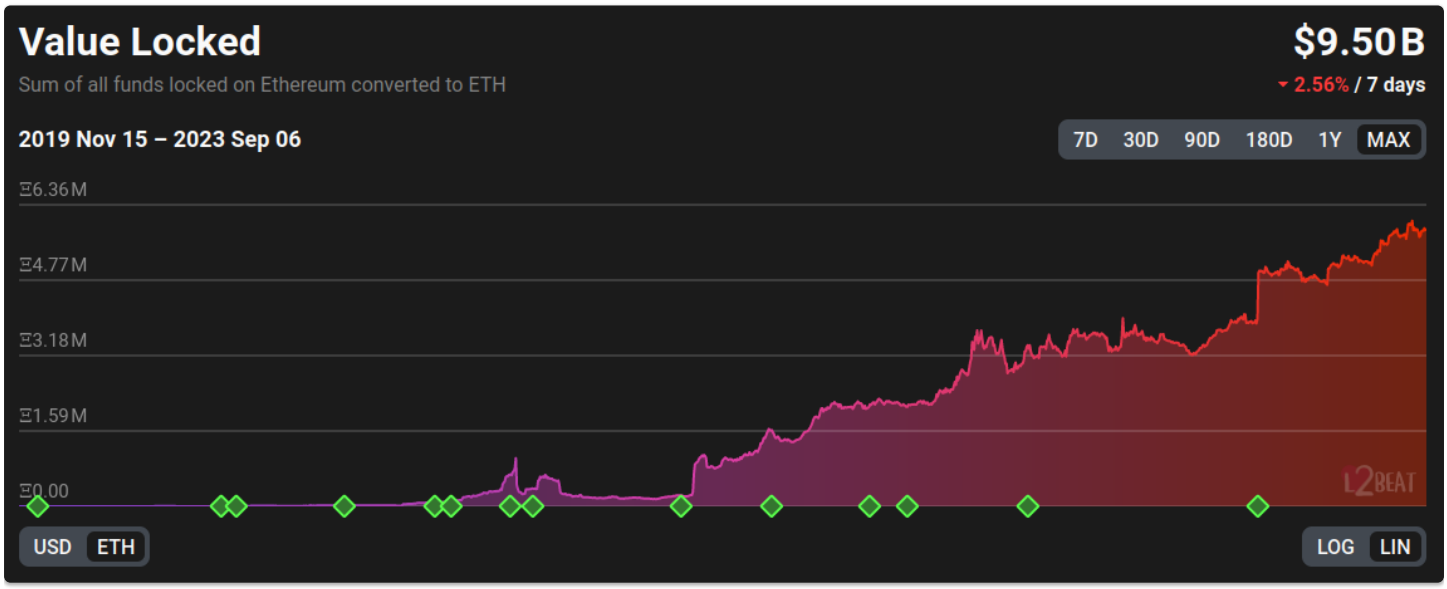
Data Availability in general

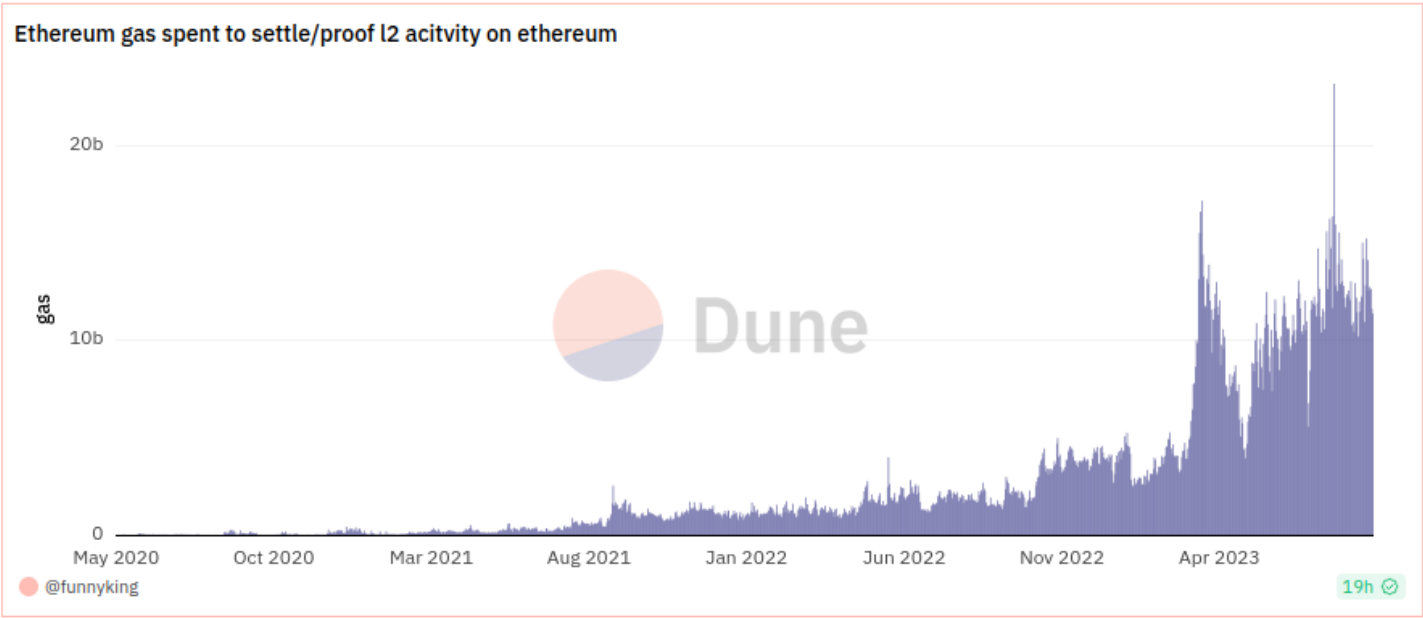
In order to re create the state, transaction data is needed, the data availability question is where this data is stored and how to make sure it is available to the participants in the system.

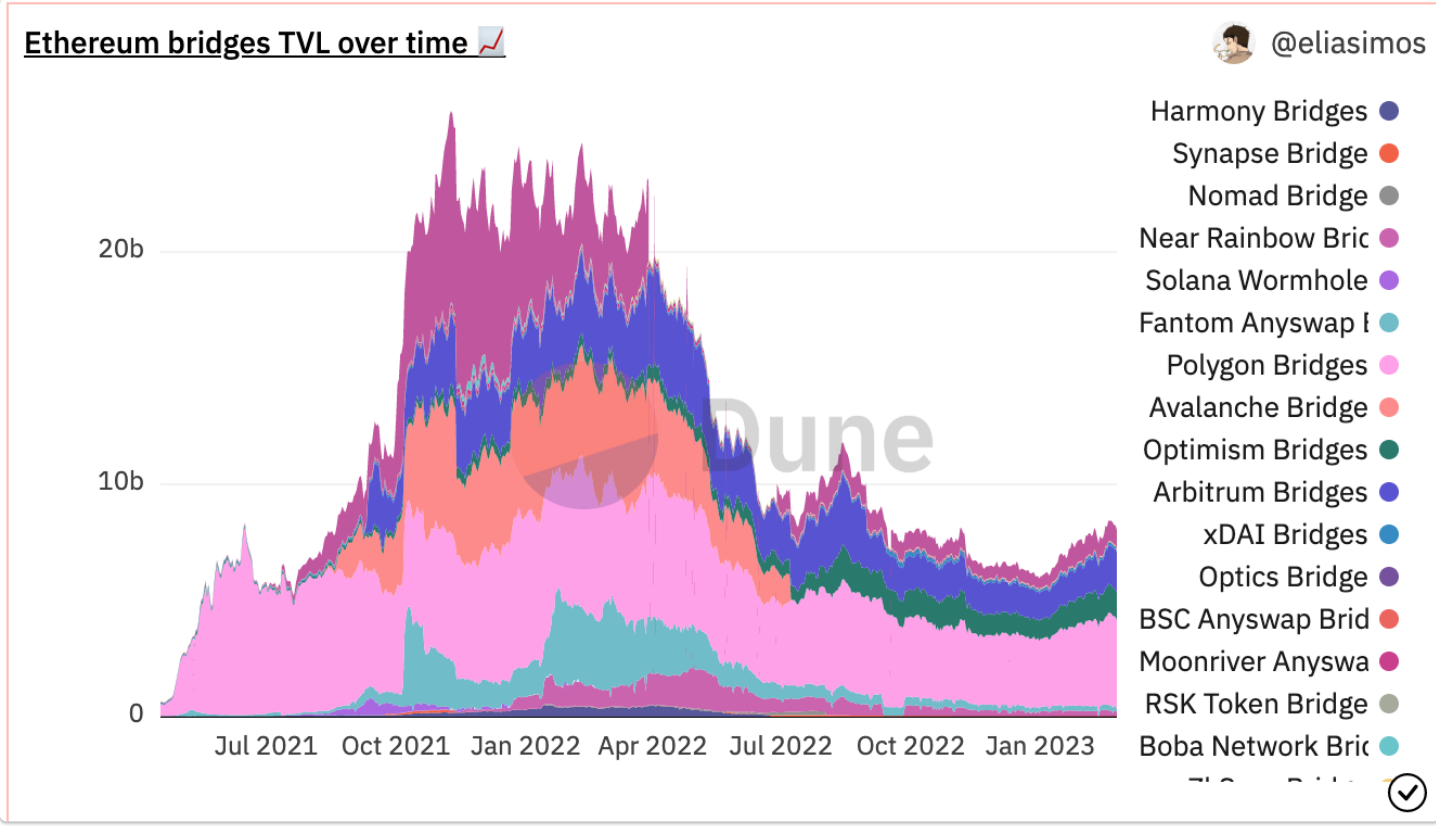
	Validity Proofs		Fault Proofs
Data On-Chain	Volition	ZK-Rollup	Optimistic Rollup
Data Off-Chain		Validium	Plasma

STARKWARE

L2 Statistics







Transaction compression and costs

For zk rollups it is expensive to verify the validity proof.
For STARKs, this costs ~5 million gas, which when aggregated is about 384 gas cost per transaction.
Due to compression techniques, the calldata cost is actually only 86 gas.
This is further reduced by the calldata EIP4488 to 16.1 gas.

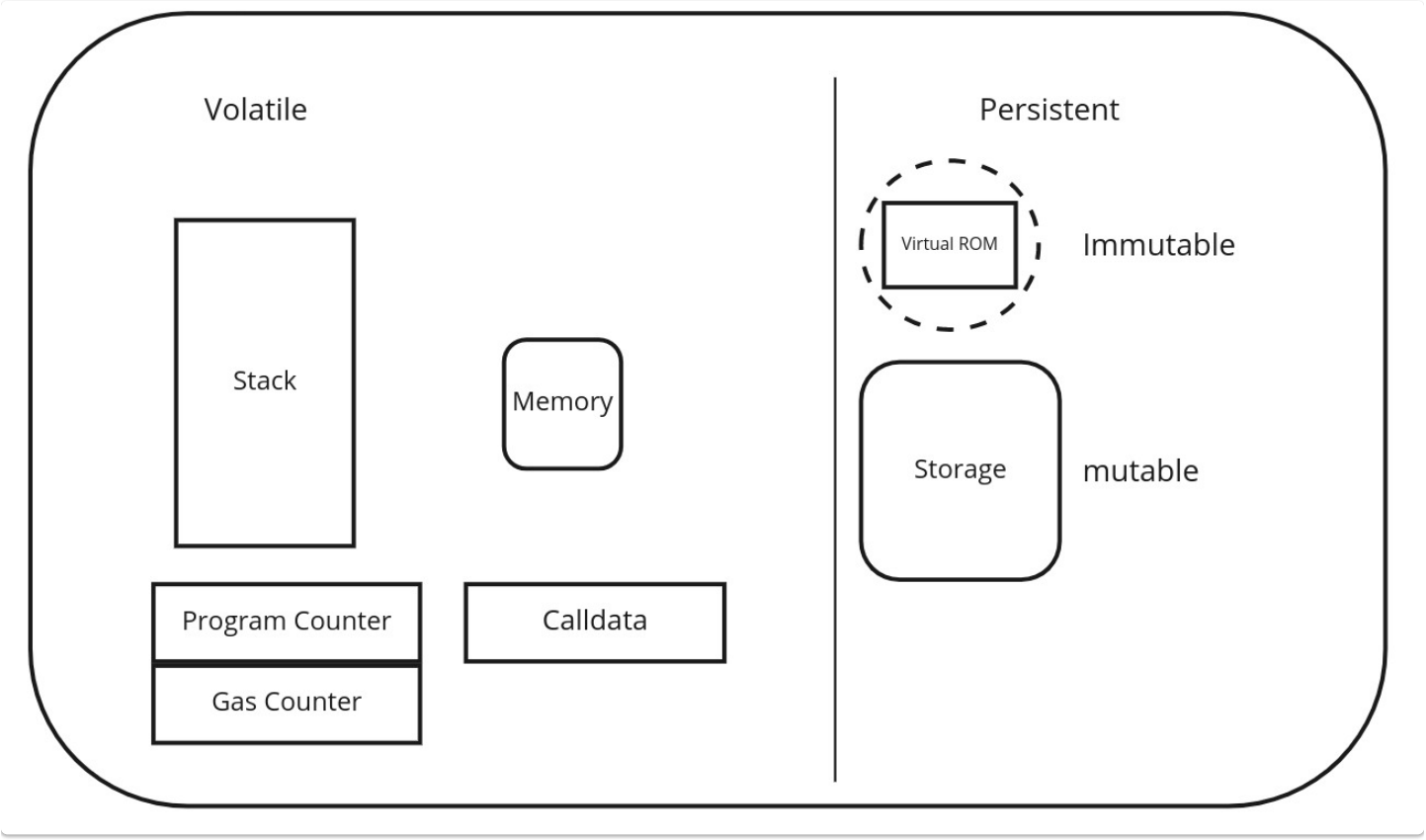
The batch cost is poly-log, so if activity increases 100x, the batch costs will decrease to only 4–5 gas per transaction, in which case the calldata reduction would have a huge impact.

At 100x the TPS today on dYdX, the total on-chain cost will reduce to only 21 gas
At this point, the bottleneck becomes prover costs for the rollup as much as on-chain gas fees, see Polygon Zero and recursive proofs

Approaches to zkRollups on Ethereum

1. Building application-specific circuit (although this can be fairly generic as in Starknet)
2. Building a universal “EVM” circuit for smart contract execution

zkEVM Solutions in general



The opcode of the EVM needs to interact with Stack, Memory, and Storage during execution. There should also be some contexts, such as gas/program counter, etc. Stack is only used for Stack access, and Memory and Storage can be accessed randomly.

AppliedZKP zkEVM

AppliedZKP divides proofs into two types:

1. State proof, used to check the correctness of the state transition in Stack/Memory/Storage.
2. EVM proof, used to check that the correct opcode is used at the correct time, the correctness of the opcode itself, the validity of the opcode, and all the abnormal conditions (such as out_of_gas) that may be encountered during the execution of the opcode.

EVM processing

In general the EVM will

1. Read elements from stack, memory or storage
2. Perform some computation on those elements
3. Write back results to stack, memory or storage

So our circuit has to model / prove this process, in particular

- The bytecode is correctly loaded from persistent storage
- The opcodes in the bytecode are executed in sequence
- Each opcode is executed correctly (following the above 3 steps)

Design challenges in designing a zkEVM

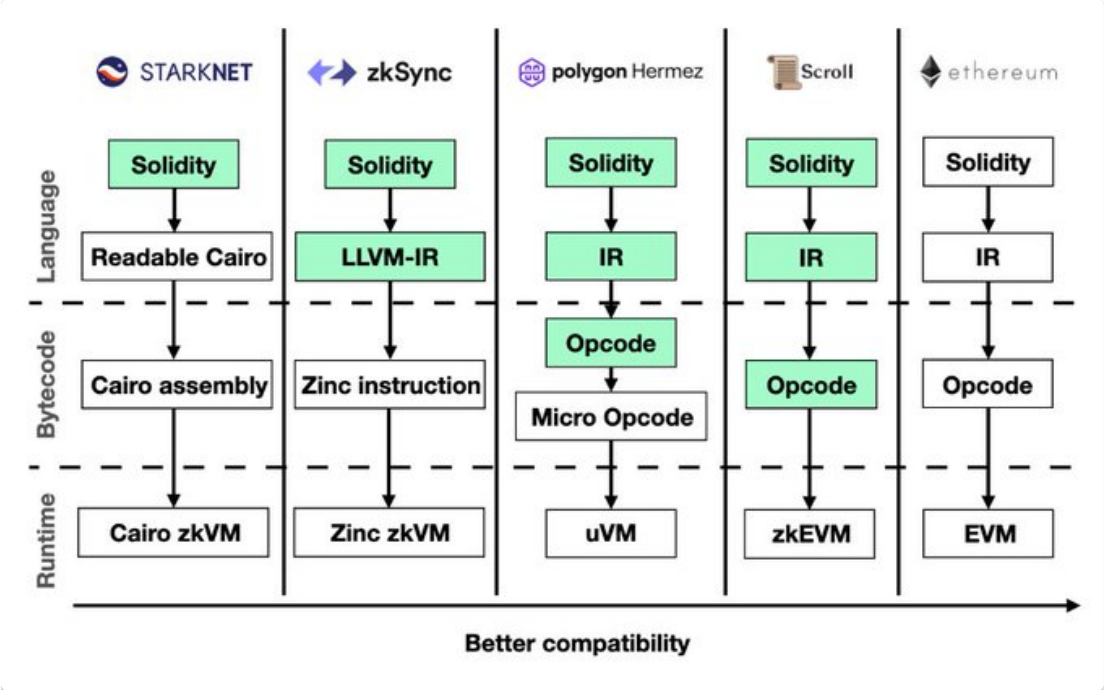
1. We are constrained by the cryptography (curves, hash functions) available on Ethereum.
2. The EVM is stack based rather than register based
3. The EVM has a 256 bit word (not a natural field element size)
4. EVM storage uses keccak and Merkle Patricia trees, which are not zkp friendly
5. We need to model the whole EVM to do a simple op code.

zkEVM taxonomy

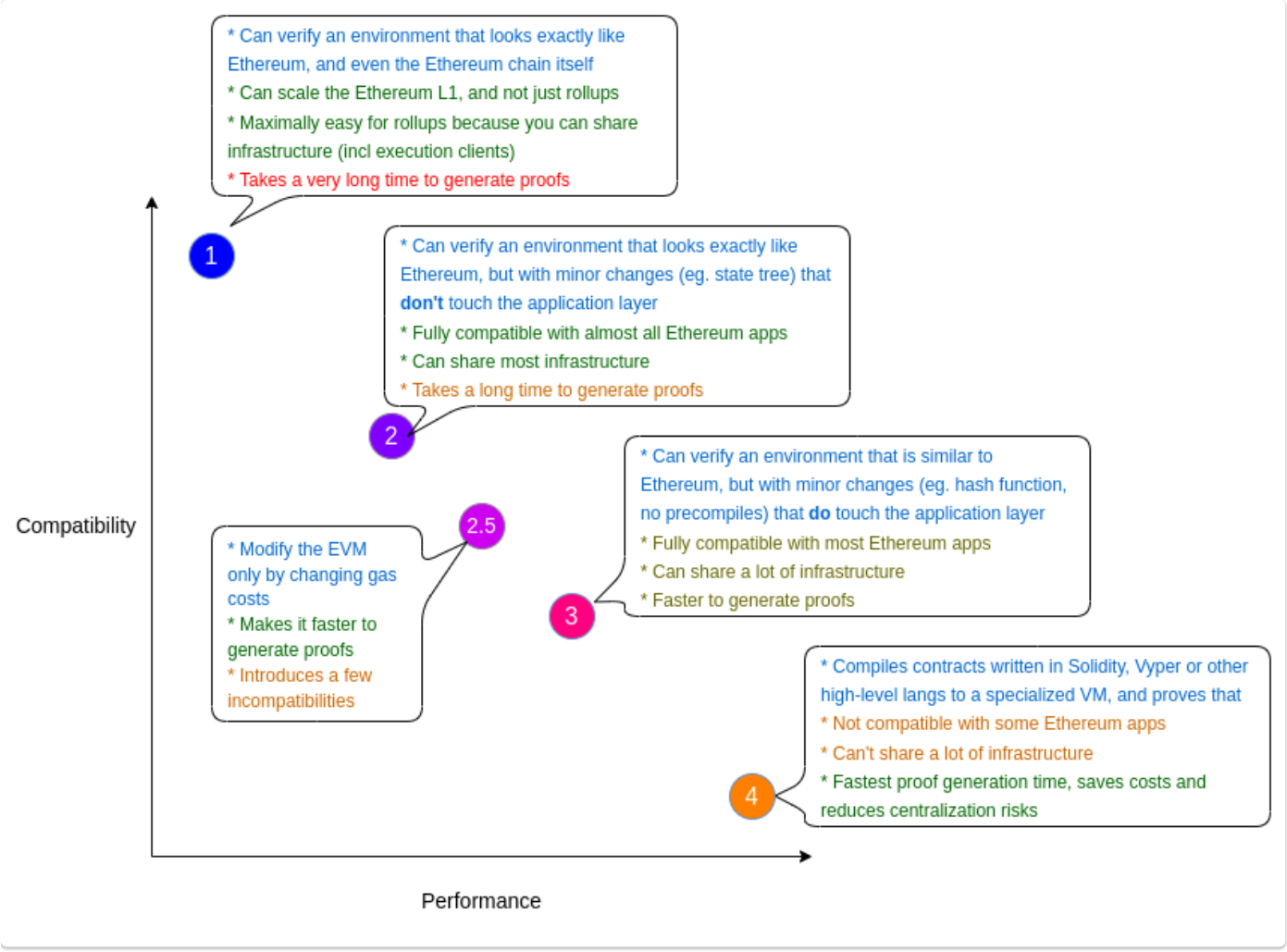
From article and article

Approaches to zkRollups on Ethereum

1. Building application-specific circuit (although this can be fairly generic as in Starknet)
2. Building a universal “EVM” circuit for smart contract execution



See Vitalik article



Type 1 (fully Ethereum-equivalent)

See zkEVM research team

Type 2 (fully EVM-equivalent)

(not quite Ethereum-equivalent)

Scroll

Hermez

Type 2.5 (EVM-equivalent, except for gas costs)

Type 3 (almost EVM-equivalent)

Scroll and Hermez in their current form

Type 4 (high-level-language equivalent)

ZKSync

Starknet + Warp

Also see

The ZK-EVM Community Edition (bootstrapped by

> community contributors including Privacy and Scaling Explorations, the Scroll team, Taiko and others) is a Tier 1 ZK-EVM.