Optimistic and Validity Rollups

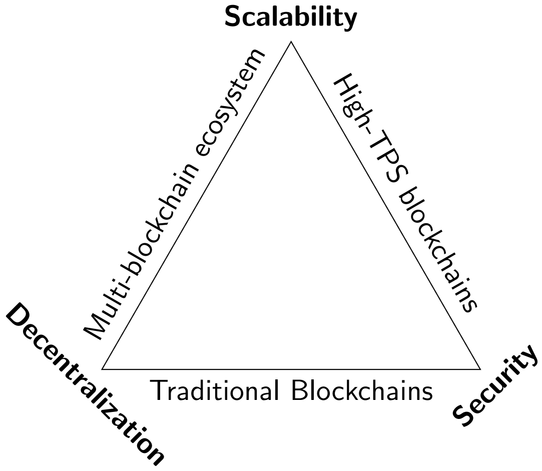
Analysis and Comparison between Optimism and StarkNet

Luca Donno

University of Bologna, Via Zamboni 33, Bologna, 40126, Italy

Scalability remains a major challenge

Most blockchains face a trade-off between scalability, decentralization and security, commonly referred to as the Scalability Trilemma.



The bottleneck

Every node in the network:

1. stores a replica of the current state of the blockchain
2. verifies consensus rules
3. executes each transaction in every block

How to solve the trilemma?

Previous attempts: state channels and Plasma.

They can't support general computation, but directionally correct:

1. move some activity off-chain
2. link on-chain activity with off-chain activity using smart contracts
3. verify on-chain what is happening off-chain

Then we got to rollups: "layer 2" blockchains that publish their blocks to another blockchain (layer 1), inheriting its consensus, data availability and security properties.

Rollups

Rollups have three main components:

- **Sequencers**: nodes that receive Rollup transactions from users and combine them into a block that is sent to layer 1. A block consists at least of a state root and the data needed to reconstruct and validate the state.
- **Rollup full nodes**: nodes that obtain, process and validate Rollup blocks from Layer 1 by verifying that the root is correct.
- **Rollup light nodes**: nodes that obtain Rollup blocks from layer 1 but do not compute the new state themselves. They verify that the new state root is valid using techniques such as fault or validity proofs.

Optimistic Rollups

Idea: optimistically accept the output of blocks without verifying their execution, but have a dispute period to challenge invalid roots using fault proofs.

First proposed in the Bitcoin whitepaper: have an “alert” system to warn light nodes about invalid blocks.

First implemented by Al-Bassam, Sonnino and Buterin: off-chain fault proof system based on error correction codes.

First optimistic rollup design: John Adler and Mikerah Quintyne-Collins in 2019.

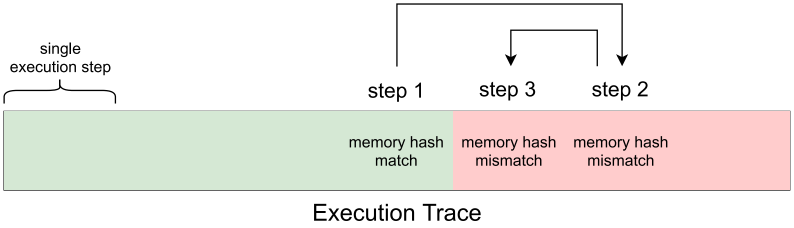
Optimism Bedrock: overview

Optimism Bedrock is a fully EVM-equivalent blockchain.

- **Deposits**: users can bridge transaction to Optimism by calling the `depositTransaction` function on the Optimism Portal smart contract. The function emits a `TransactionDeposited` event that is used by each rollup node to create an L2 transaction.
- **Sequencing**: users can also directly send transactions on the L2. The sequencer get this transactions and build blocks to send to L1. Nodes can fully reconstruct the rollup state from this data.
- **Withdrawals**: the L1 needs to know the state of L2 to accept L2-to-L1 messages. An L2 Proposer publishes the state root for each L2 block on L1. These roots are optimistically accepted as valid if no fault proof is performed during the dispute period.

Optimism Bedrock: the fault proof system

A binary search is performed between the Proposer and the Challenger to find the first instruction of disagreement in the execution trace of the block, which is executed on-chain on a MIPS interpreter. The system is economically incentivized.



Validity Rollups

The goal of a Validity Rollup is to cryptographically prove the validity of the state transition given the sequence of transactions with a short proof that can be verified sub-linearly compared to the time of the original computation.

These kind of certificates are called *computationally integrity proofs* and are practically implemented using SNARKs, which use arithmetic circuits as their computational model.

STARKs are a kind of SNARK that do not require a trusted setup and are quantum resistant, while giving up some efficiency on proving and verification compared to other solutions.

StarkNet: overview

StarkNet is a L2 blockchain based on the Cairo VM.

- **Deposits**: users can bridge transaction to StarkNet by calling `sendMessageToL2` function on the bridge contract. The function emits a `LogMessageToL2` event that is used by each rollup node to create an L2 transaction.
- **Sequencing**: users can directly send transactions on the L2. The sequencer get this transactions, build a block, calculate the new state and publishes its state root on L1 as well as the state difference between the current and the previous block.
- **Withdrawals**: the L1 needs to know the state of L2 to accept L2-to-L1 messages. The Prover sends a STARK proof on L1 to a smart contract verifier to prove that the state root is correct.

StarkNet: STARK proofs

Instead of creating an arithmetic circuit for every smart contract, StarkNet just implements the circuit that represents the FDE cycle of a von Neumann architecture. In this way the number of constraints is fixed, allowing only one verifier program for every program whose computation needs to be proved.

StarkNet uses the shared prover SHARP to aggregate multiple transactions into a single STARK proof. The sub-linear cost of verifying a validity proof allows its cost to be amortized over multiple transactions.

Comparison: withdrawal time (1/2)

The withdrawal delay depends on the frequency on which state roots are posted on L1.

A state root within an optimistic rollup is finalized after the dispute period (usually around 7 days). The length of the dispute period is determined by the fact that fault proofs can be censored until its end:

$$\mathbb{E}[\text{subtracted value}] = Vp^n$$

The system is made secure by asking root proposers to stake a much larger amount of value than this expected value.

In the paper we propose a smart contract that incentivize block producers to perform a censorship attack on fault proofs by using bribes greater than the amount of fees that they would get from including the transaction.

Comparison: withdrawal time (2/2)

Withdrawals from a validity rollups are possible as soon as a state root is published on L1 and proven.

It can be as fast as every block, but in practice there is a trade-off between root finalization speed and proof aggregation.

Starknet currently provides validity proofs for verification every 10 hours, but this delay is intended to be decreased as transaction activity increases.

Comparison: transaction costs

Optimism posts L2 transaction calldata on L1 as calldata, while StarkNet posts state's storage differences on L1 as calldata.

A gas compression rate for Optimism can be calculated by comparing the amount of gas used on L2 in a given period of time and the amount of gas spent on L1. With this method a gas compression rate of $\sim 20 : 1$ is found.

Since StarkNet does not use the EVM the transaction cost compression cannot be easily estimated. By assuming the cost of execution and calldata to be negligible, it is possible to calculate the compression ratio of storage writes compared to L1. By using this method a storage write compression rate of $\sim 24 : 1$ is found.

Further optimizations (1/2)

We propose further optimization for both optimistic and validity rollups.

```
contract AddressCache {
    mapping(address => uint32) public address2key;
    address[] public key2address;

    function cacheWrite(address _address) internal returns (uint32) {
        require(key2address.length < type(uint32).max, "AddressCache: cache is full");
        require(address2key[_address] == 0, "AddressCache: address already cached");
        // keys must start from 1 because 0 means "not found"
        uint32 key = uint32(key2address.length + 1);
        address2key[_address] = key;
        key2address.push(_address);
        return key;
    }

    function cacheRead(uint32 _key) public view returns (address) {
        require(_key <= key2address.length && _key > 0, "AddressCache: key not found");
        return key2address[_key - 1];
    }
}
```

Figure: Calldata optimization for optimistic rollups.

Further optimizations (2/2)

Storage writes can be optimized using Bloom's filters.

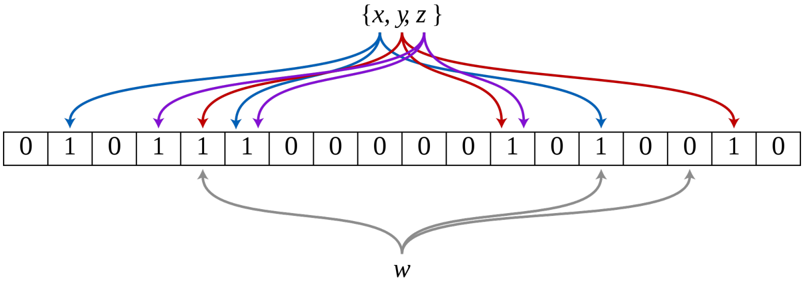


Figure: Storage optimization for validity rollups.

Ethereum compatibility

Optimism is EVM-compatible, meaning that all Ethereum smart contract can be redeployed without any modification nor new audits. Wallets remain compatible, as well as dev tools, security tools, indexing tools and oracles.

StarkNet decided to adopt a different VM to accommodate validity proofs. This implies rebuilding an entire ecosystem, with the advantage of a greater freedom. StarkNet natively implements *account abstraction*, which has also been proposed for Ethereum in 2020 but it is still yet to be implemented.

Another important security aspect is client diversity: a bug in the implementation of the fault or validity proof system affects safety. This can be prevented with a wider client diversity, which is easier for EVM-based rollups.

Conclusion

The choice of developing either an optimistic or a validity rollups is mainly shown as a trade-off between complexity and agility.

Optimism, though, possess a modular structure that allows it to become a validity rollup in the future: instead of proving fault proofs for a MIPS machine, it is possible to prove validity proofs for it. RISC Zero is a verifiable micro-architecture with STARK proofs already in development based on RISC-V that can be used in alternative to MIPS.