BLOCKCHAIN SCALABILITY

Group 3

MASRI, SOUSA, PEROTTI, PICERNO, SEIGLE

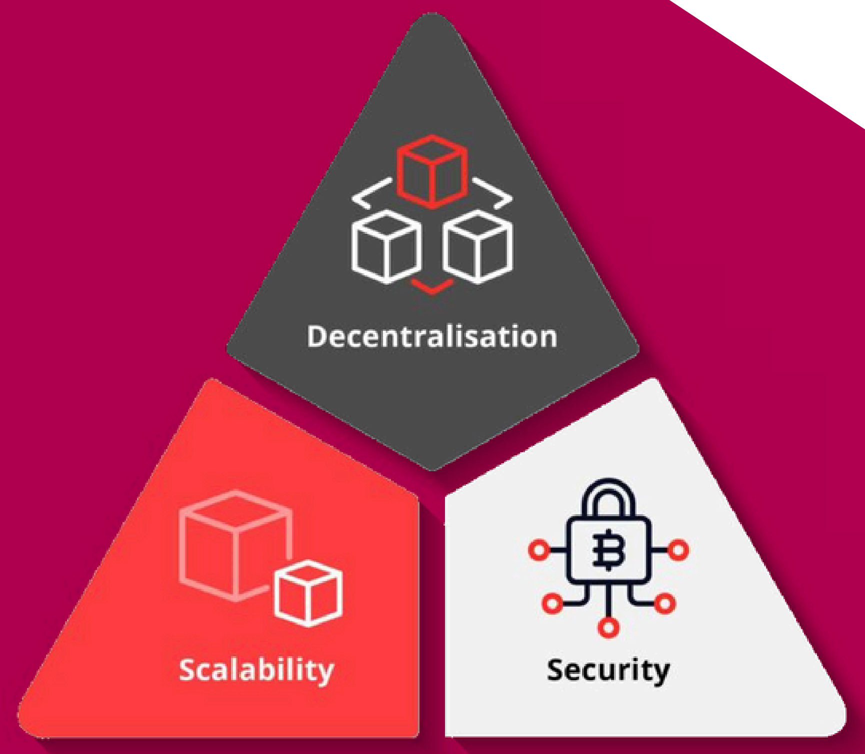
Outline

1. **Introduction**
2. **Scalability Layer 1**
3. **Introduction to Ethereum and L2**
4. **Optimistic Rollups**
5. **Zero-Knowledge Rollups**
6. **Comparison**
7. **Conclusion**

Motivation

The fundamental problem

- Ethereum: only 15–25 TPS vs. thousands of requests for mass applications
- High gas fees during periods of congestion



Two proposed approaches

- **Layer 1:** sharding, new consensus protocols
- **Layer 2:** rollups

Scalability

Depends on the Consensus Protocol :

- PoW
- PoS
- PoH

On ETH and SOL, Smart Contracts can be implemented. They open the door for scalability.

	3-7 transactions per sec.	**PoW**
	15-25 transactions per sec.	**PoS**
	2000-4000 transactions per sec.	**PoS, PoH**

Scalability Level 1 - Sharding

General idea

- Splitting the network into sub-networks (shards)
- Asking each sub-network to process and mine fractions separately
- Reassemble the fractions to build the block

Advantages

1. Greater scalability
2. Increased throughput thanks to parallel processing.

Challenges

1. Complex management of cross-shard transactions.
2. Risks to security and consistency.

Rollups

Definition

Layer 2 scalability solutions that process transactions off-chain and publish on the main blockchain a compressed representation or a proof of correctness

Goal

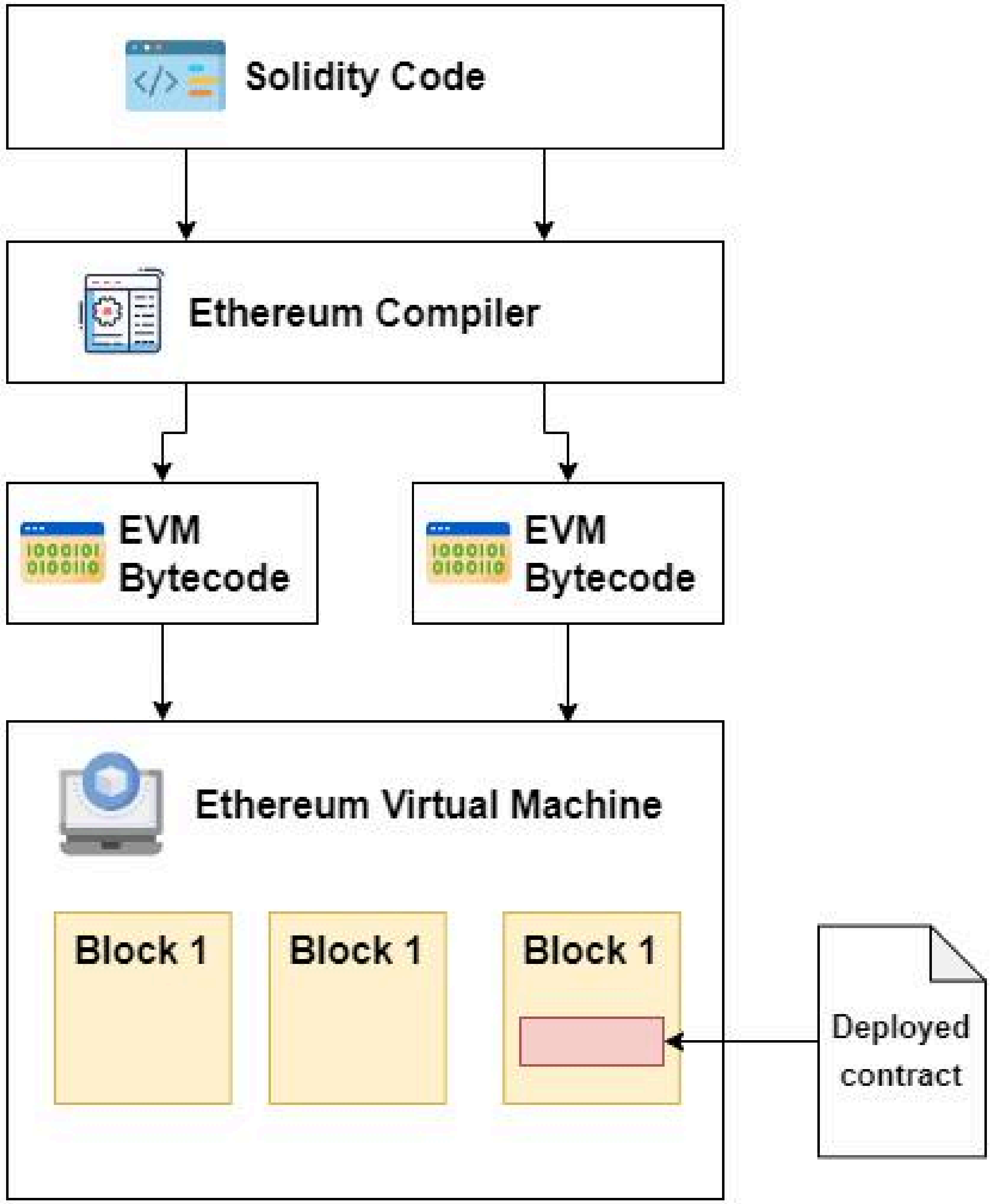
Improve scalability and reduce transaction costs by reducing workload on Layer 1, while preserving the security guaranteed by this last one.

Main types

- Optimistic Rollups: assume validity by default, verify only in case of dispute (fraud proof)
- ZK-Rollups: generate a cryptographic validity proof for each transaction batch

Ethereum - Introduction

- Ethereum is the open source second biggest cryptocoin
- It was designed as a **programmable blockchain,** not limited to value transfers
- It uses smart contracts, self-executing programs that can manage complex on-chain state
- It includes the Ethereum Virtual Machine (EVM), which can execute arbitrary deterministic logic.
- The protocol is designed to accept and process different data types, not only transactions.



Ethereum - Smart Contracts

Main Characteristics

- Self-executing programs
- They define logic, rules, and state transitions that **all nodes must validate.**
- They can store and update on-chain state, enabling **complex applications** beyond simple transfers.
- Because they run in the **Ethereum Virtual Machine** (EVM), their execution is standardized across all nodes.
- Smart contracts **can verify external data** (proofs, signatures, commitments) and update state accordingly.
- They can act as **on-chain arbiters**, deciding whether submitted data is valid or fraudulent.

Ethereum - Perfect Scenario for L2

"Layer-2 refers to any off-chain network, system, or technology built on top of a blockchain (commonly known as a Layer-1 network) that strives to extend the capabilities of the underlying base layer network."

How to implement them

- A Layer 2 runs off-chain computations or transactions but connects to Ethereum via smart contracts on the base layer.
- The L1 smart contract acts as the trust anchor, storing state commitments or cryptographic proofs from the L2.
- By programming rules correctly, the L2 can update balances, records, or contract states off-chain while keeping the main chain as the final arbiter.
- Essentially, smart contracts **bridge** the L2 and L1.

Optimistic Rollups - How They Work

Main Characteristics

1. Transactions are processed **off-chain**, while compressed data is published on Ethereum as calldata or blobs.
2. The term **"optimistic"** derives from the default assumption of validity: no cryptographic proofs are provided immediately.
3. Operators (sequencers) group transactions into batches, reducing user costs through data compression.
4. After publication on L1, a **challenge period** begins: anyone may dispute the state by submitting a fraud proof.
5. In case of fraud, the disputed transactions are re-executed on Ethereum and the malicious operator suffers **slashing**.

Consequence

Rollup blocks are accepted as valid only if uncontested by the end of the challenge period.

Optimistic Rollups - State Management

State Commitments

1. The L2 state (accounts, balances, smart contracts) is organized in a **Merkle Tree** called the state tree.
2. The **state root** represents the final rollup state and is stored in the Ethereum contract.
3. For each batch, the operator must publish both the **old** and the **new** state root.
4. The operator also computes the **Merkle root of the batch**, enabling inclusion proofs (Merkle proofs) for individual transactions.

Guarantee

The state commitment allows the Rollup contract to validate and, if necessary, invalidate incorrect roots, restoring the correct chain state.

Optimistic Rollups - Data Availability

State Commitments

- Transaction data is published on Ethereum in compressed form.
- Two main options:
  - **Calldata**: immutable and non-persistent area stored in blockchain history.
  - **Blobs**: similar to calldata but removed after about 18 days; reduce the costs further.
- Publication on L1 ensures anyone can reconstruct the L2 state without relying on the operator.

Implication

Improves transparency, censorship resistance and decentralization, but increases costs compared to pure off-chain solutions.

Optimistic Rollups - Fraud Proving

How it works

1. The operator publishes an assertion (batch + state root).
2. During the challenge period anyone may dispute it.
3. If disputed → execution of a fraud proof.

Proving

- **Single-round**: full re-execution on L1.
- **Multi-round**: dispute split into multiple steps, lower costs.

Incentives

- Operators locked by a **bond**.
- **Slashing** in case of fraud.
- Security guaranteed if at least one honest node exists.

Note Ensures trustless security but introduces delays due to the challenge period.

ZK Rollups - How They work

Main Characteristics

1. **Off-chain execution:** transactions are processed by a sequencer or PoS validators.
2. **Publication on L1:** the operator submits the updated state root along with a validity proof
3. **Data availability:** compressed data is published as calldata or via cryptographic commitments (batch root).
4. **On-chain verification:** the rollup smart contract validates the proof before updating the state.
5. **Immediate finality:** once the proof is verified, transactions are final without a challenge period.

Consequence

ZK validity proofs eliminate the need for honest nodes: correctness is guaranteed cryptographically in a trustless way.

ZK Rollups - State Management

State Commitments

- The L2 state is represented by a **Merkle Tree**, whose **state root** is stored on-chain
- Each batch generates a new state root that replaces the previous one after proof verification.
- A **batch root** is also computed, representing the Merkle root of all transactions in the batch.
- The batch root allows proving inclusion of a transaction via Merkle proof without downloading the entire batch.

Advantage

The combination of state root and batch root enables secure and lightweight verification, reducing the amount of data processed on L1.

ZK Rollups - Data Availability

Data publication

- Aggregated transaction data is published on Ethereum to guarantee **data availability**.
- Usually sent as **calldata** or as **cryptographic commitments** (e.g., batch root).
- This allows anyone to reconstruct the L2 state without trusting the sequencer.
- Publishing data on L1 increases security and censorship resistance.

Note

Publication reduces the risks of state loss but implies gas costs and requires compression to keep fees sustainable.

ZK Rollups - Validity proofs

General Idea: Each batch includes a validity proof verified by the L1 smart contract → the new state root is accepted only if the proof is valid.

Characteristics

1. State updates are **immediately final.**
2. No challenge period needed.
3. Withdrawals without delays.

Challenges

1. Proof generation is expensive and complex.
2. Specialized hardware required.

Advantage

Security based on trustless cryptographic proofs, independent of operator honesty.

Optimistic vs ZK Rollups - Summary

Category	Optimistic	ZK
Validation	Default valid; *fraud proof* in case of dispute	Each batch with cryptographic *validity proof*
Finality	Delayed (challenge $\approx$ 7 days)	Immediate, after proof verification
Security	Incentives + honest nodes	Trustless cryptography
Computation	Lightweight, no special hardware	Costly proving, dedicated hardware
Operations	EVM compatible, usually centralized sequencer	Often centralized sequencer, more capital efficient

Conclusions

- Both Rollups are **key solutions** for fighting the scalability problem
- **Optimistic:** EVM compatibility, incentive-based security, but slow finality
- **ZK:** immediate finality, cryptographic security, higher computational costs
- Future: integration of multiple solutions. Sharding + rollup evolutions

Summary

Choosing between the different solutions means making a trade-off between **speed**, **costs** and **computational complexity**

Thanks For Your Attention!
Questions?