Scaling Blockchains and the Case for Ethereum

Aditya Asgaonkar

Abstract This chapter provides a high-level introduction to scaling solutions for blockchains, with a special focus on Ethereum 2.0. Current blockchain capacity is a hurdle for the widespread adoption of Web3 and cryptocurrencies. First, we discuss the considerations and pitfalls of blockchain scaling strategies. We then explore the design landscape—layer-1 and layer-2 solutions—and discuss concepts in each category, namely sharding, rollups, and sidechains.

1 Introduction to the Scaling Problem

The two most popular blockchains—Bitcoin and Ethereum—are currently able to support between 5 and 15 transactions per second (TPS). With increasing mainstream Web3 adoption through Decentralized Finance (DeFi), Non-Fungible Tokens (NFTs), Decentralized Autonomous Organizations (DAOs), etc., blockchain scalability is becoming an increasingly important problem to solve. Ethereum has seen an explosive increase in usage since 2020, and at times network congestion has led to exorbitantly high transaction fees. Scaling blockchain capacity is a prerequisite for the widespread and commonplace adoption of Web3 systems.

This chapter will be focusing on scaling solutions for public, permissionless, and general-purpose blockchain systems. Let us define these terms:

Public: The blockchain can be used by the general public.

Permissionless: The requirements to participate in the decision-making process are defined by the protocol and are accessible to the general public. There is no gatekeeper entity that chooses the participants—the protocol admits all actors wishing to participate that satisfy the requirement.

General Purpose: The blockchain utilizes a general-purpose transaction system that supports smart contracts, e.g., the Turing-complete Ethereum Virtual Machine [1].

A. Asgaonkar (✉)
Ethereum Foundation, Bern, Switzerland
e-mail: aditya@ethereum.org

D. A. Tran et al. (eds.), *Handbook on Blockchain*, Springer Optimization and Its Applications 194, https://doi.org/10.1007/978-3-031-07535-3_6

1.1 Considerations

The main considerations while evaluating blockchain scaling solutions are:

1. Throughput and Latency;
2. Decentralization; and
3. Security.

Throughput and Latency

Throughput is the number of transactions per unit time that the blockchain system forms consensus over. Latency is the time required for a transaction to become a part of the chain under consensus. A higher throughput and a lower latency are desirable for any transaction system.

Decentralization

A key aspect of blockchain systems is decentralization. Qualitatively, the factors contributing to decentralization of a blockchain are:

- **Participation**: the participants of the decision-making process are not concentrated in a single group.
- **Verification**: a larger number of users are able to verify the output of the blockchain.

Nodes that verify all blocks and transactions are called *full nodes*. If the throughput of the blockchain is increased and full nodes have to verify a larger number of transactions at the same time, the minimum hardware requirements for operating a full node will be increased. This reduces the number of people that are able to run full nodes and verify the chain output, hence reducing the decentralization of the chain.

Security

The security of a blockchain system is quantified by the fraction of nodes that must misbehave to cause a safety or liveness failure of the decision-making process. Security is affected by the model of consensus employed, which involves factors such as:

- Honesty Assumptions: The assumptions that are made about the behavior of participants, e.g., unconditionally honest (actors that follow the protocol unconditionally), economically rational (actors that are willing to deviate from the protocol if profitable), etc.

- Quality of Safety: Consensus protocols differ in their guarantees about safety. Traditional BFT protocols provide a safety threshold—a minimum number of protocol-violating nodes required to cause a safety violation. In the context of blockchain consensus, it may be beneficial to consider the stricter notion of *accountable safety* threshold [11]—a minimum number of protocol-violating nodes that can be held accountable in a provable manner in case of a safety violation.

1.2 Naive Scaling Solutions

Bigger/Faster Blocks

A common naive solution to scaling a blockchain protocol is to increase the size and/or frequency of blocks. This has an effect on increasing the throughput and/or decreasing latency of the system. For example, if a proof-of-work blockchain increases the size of its blocks[1] by a factor of k, then the throughput of the system increases by k. However, this is not without tradeoffs. The minimum hardware requirements for verifying blocks will increase—the slowest processor that can verify the blockchain will need to be k times as fast as the earlier one. This reduces decentralization by limiting the number of nodes that can verify the chain and participate in the network. If the block size becomes huge, then only very large servers will be able to verify the chain.

1.3 Types of Scaling Solutions

Scaling solutions can be broadly classified into two categories:

- Layer-1: Scaling is achieved by employing fundamentally different architectures for the blockchain protocol. This includes changes to the consensus mechanism, network architecture, distribution of verification duties to subsets of the network, etc.
- Layer-2: Scaling is achieved by designing a transaction system such that a large number of transactions in the new system are executed with a small number of transactions on the base blockchain's transaction system. Such systems are designed for users to interact with the existing underlying blockchain, and rely on the security of the underlying blockchain protocol.

[1] In Ethereum, block size is defined by the *gas limit*—a limit on the total computing operations carried out by transactions included in a block.

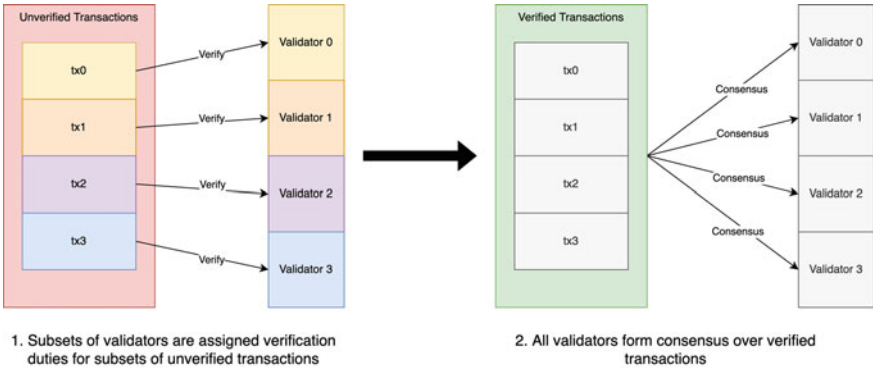


Fig. 1 Sharding designs improve throughput by distributing transaction verification tasks among subsets of validators and retain security by forming consensus over all transactions with the entire validator set

2 Layer-1 Scaling Solutions

2.1 Sharding

A popular layer-1 scaling solution is *sharding*, wherein the verification duties for the blockchain's transaction system are distributed among multiple smaller subsets of the participants, but consensus is formed by the entire set. Most sharded blockchains employ proof-of-stake[2] consensus in their design. Participants are called *validators* (Fig. 1).

2.2 Ethereum 2.0

Beacon Chain and Shard Chains

- **Beacon Chain**: The beacon chain is responsible for forming consensus over shard chain blocks and bookkeeping related to the consensus process (Fig. 2).
- **Shard Chains**: Shard chains are where the users' Ethereum transactions are executed. Each shard chain has an independent state and is responsible for validating and executing transactions concerning that piece of state.

[2] Proof-of-stake is a blockchain design that relies on intrinsic resources (such as its own cryptocurrency) to act as a mechanism to choose the consensus participant set—e.g., Ethereum 2.0 requires validators to deposit a certain minimum amount of Ether in order to be included as a consensus participant. In contrast, proof of work relies on some extrinsic resource to choose the participant set—e.g., Bitcoin limits its participant set to people with computing resources, and the probability of contributing to the chain is dependent on the speed and capacity of the computing resource.

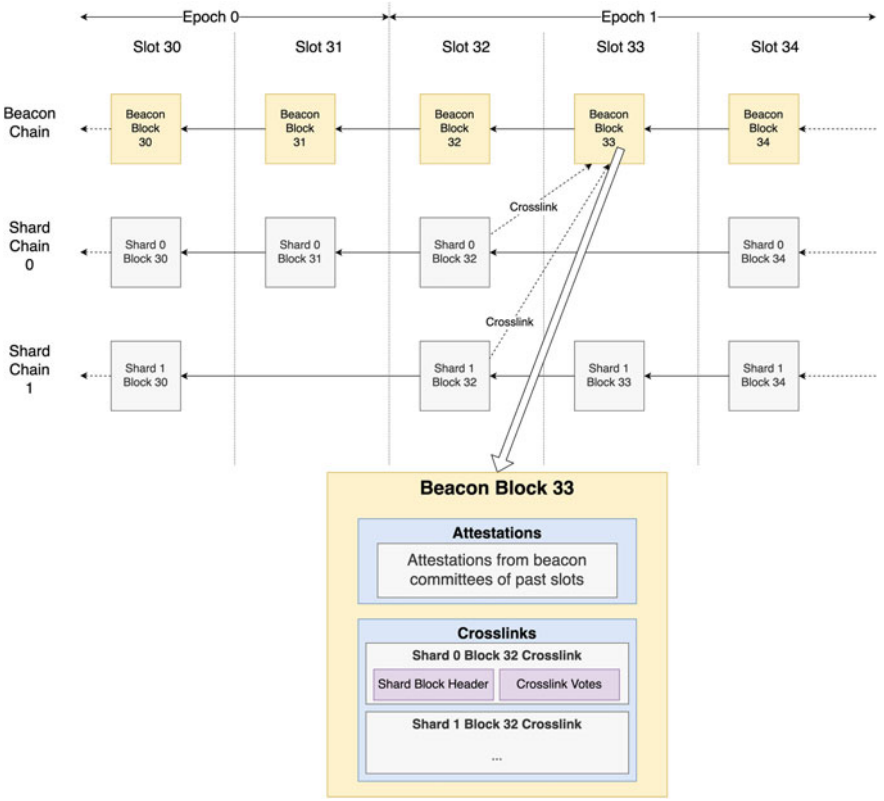


Fig. 2 Beacon chain, shard chains, and crosslinks

! Note

The design of shard chains is a work-in-progress effort. The most up-to-date design can be found in the Ethereum 2.0 specifications repository on GitHub: https://github.com/ethereum/consensus-specs/tree/dev/specs/sharding.

- **Slots and Epochs**: Time is divided into epochs, which are further divided into 32 slots. The current parameters are configured to 12 s per slot.
- **Validators**: Participants of Eth2's consensus process are called *validators*. Validators have two main duties:
  - Verifying and finalizing[3] beacon blocks
  - Verifying shard blocks

[3] A block is finalized when the validators decide using the consensus process that the block is a part of the canonical chain, and this block cannot be reverted in the future. Finalization is the consensus process to arrive at this decision.

- **Proposers**: At every slot in every chain (i.e., beacon chain and shard chains), a randomly chosen validator is assigned to be the block proposer in that chain. The proposer packages attestations[4] seen from other validators into a block, builds the new block on top of the head of the chain and gossips the new block in the p2p network.
- **Committees**: Committees are groups of validators that are assigned a duty. Based on the type of duty assigned, there are two types of committees:
  - **Beacon committees**: For every epoch, the entire validator set is divided into beacon committees such that there is one beacon committee assigned for every slot. At its assigned slot, each validator in the beacon committee makes an attestation for a beacon block. A validator's attestation for a particular beacon block indicates that the validator has verified the block, and constitutes a vote for the beacon block to be considered in the consensus process.
  - **Shard committees**: Similar to beacon committees, the entire validator set is divided such that there is a shard committee for some shard(s) in every slot. At its assigned slot, each validator in the shard committee makes a crosslink vote for a block in that shard. The crosslink vote indicates that the validator has verified the shard block, and should be *crosslinked* into the beacon chain.
- **Crosslinks**: A shard block that has crosslink votes from more than two-thirds of a shard committee can be crosslinked into the beacon chain—the shard block header is included in a beacon block, and the shard block is finalized when that beacon block gets finalized.

Beacon chain blocks contain attestations and crosslink information (i.e., shard block headers and corresponding crosslinks votes). Shard chain blocks contain user transactions, similar to Eth1 blocks today.

Consensus Process

Casper the Friendly Finality Gadget (FFG) [10] is a major component[5] of the Eth2.0 consensus process, and provides the following guarantees:

- **Accountable Safety**: If two conflicting blocks are finalized, then at least one-third of validators have broken the Casper FFG rules, and these validators can be identified.
- **Plausible Liveness**: In any state of the protocol, a deadlock is impossible and the validators can make new votes that progress the protocol (i.e., finalize a new block) without violating any Casper FFG rules.

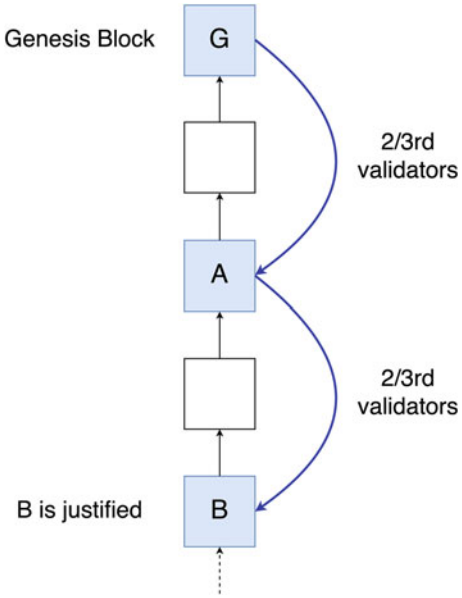
[4] An attestation is a vote for a block from a validator and is used in the consensus process to finalize the block.

[5] Casper FFG in itself is not a full consensus protocol. It provides the rules for identifying when a state of consensus has been reached (the *finalization rule*) but does not describe how to achieve such a state. In this context, it is called a finality gadget.

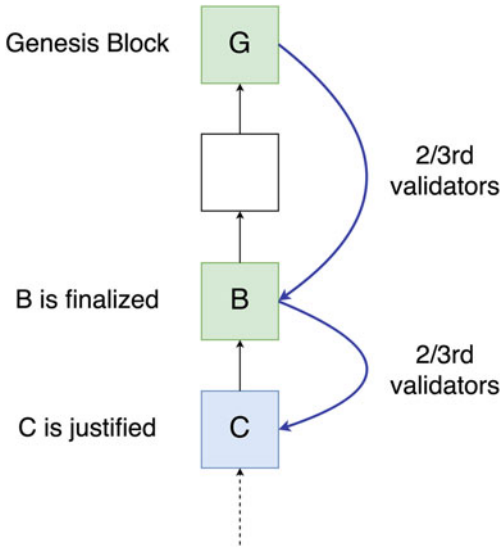
Casper FFG

A brief description of the Casper FFG mechanism:

- **Justified Block**: A block is justified if it is the genesis block, or more than two-third of validators have made votes (A, B), where A is some ancestor of B and A is a justified block.



- **Finalized Block**: A block is finalized if it is the genesis block, or B is justified and more than two-third of validators have made votes (B, C), where C is the direct child of B (i.e., $height(C) = height(B) + 1$)



Only blocks at epoch boundaries are considered for Casper FFG in the beacon chain, for two reasons:

- Consensus processing is done at the interval of an epoch rather than every slot
- It allows for the entire validator set to communicate their consensus votes over the length of an epoch, which reduces p2p network congestion as compared to all validators communicating their votes in the same slot.

Fork Choice Rule

The fork choice rule describes how a canonical chain is chosen from a set of blocks which contain multiple different chains. Output from the Casper FFG mechanism is not enough to choose a canonical chain from a block tree—justification or finalization requires votes from the entire validator set, and only a sample of the validator set is heard from at each slot. Therefore, there is a need for an algorithm to choose a canonical chain from the unfinalized section of the block tree. The beacon chain uses the Hybrid Latest Message-Driven (LMD) GHOST fork choice rule, as described in Algorithm 3 (Fig. 3).

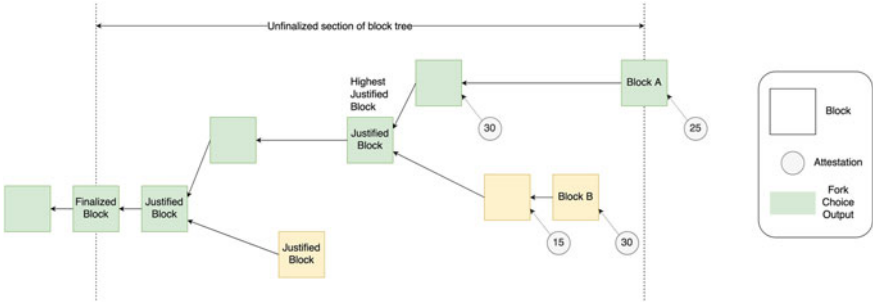


Fig. 3 Hybrid LMD GHOST chooses the chain defined by Block A in the above scenario

Fork Choice Algorithm

Algorithm 1 LMD GHOST Score

```
procedure LMD_GHOST_SCORE(b)
    M ← list of latest attestations from all validators
    score ← number of attestations in M voting for b
    for all child c of b do
        score ← score + LMD_GHOST_SCORE(c)
    end for
    return score
end procedure
```

Algorithm 2 LMD GHOST Fork Choice

```
procedure LMD_GHOST(b)
    while b has children do
        b ← arg max_{c child of b} LMD_GHOST_SCORE(c)
    end while
    return b
end procedure
```

Algorithm 3 Hybrid LMD GHOST Fork Choice

```
procedure HYBRID_LMD_GHOST(C)
    b ← highest justified block in chain C
    return LMD_GHOST(b)
end procedure
```

Scalability Analysis

First, let us define some notation. Let

- c be the computations per second of a single node,
- n be the number of participants in consensus,
- *verification* (c) denote the computation required for verifying c blocks, and
- *consensus* (n, c) denote the computation required at each node for n participants and c blocks.

The verification task involves verifying digital signatures and executing transactions, i.e., looking up and operating on pieces of state. The *verification* function is assumed to be linear in the number of blocks to be verified.[6]

To come to consensus on a larger number of blocks with each consensus instance taking in a fixed amount of data, the number of times that the consensus process is run needs to be increased proportionally to the increase in blocks. So, the $consensus$ function is assumed to be linear in the number of blocks.

In a single proof-of-stake chain with n participants, if the chain has a throughput of c blocks per second, then each node is performing $verification(c) + consensus(n, c)$ computations per second. If the rate of processing at each node becomes p times, each node can process $p \cdot (verification(c) + consensus(n, c)) = verification(p \cdot c) + consensus(n, p \cdot c)$ computations per second, i.e., the chain throughput becomes $p * c$ blocks per second.

In the Eth2 sharded proof-of-stake system with n participants and s shards, each having a throughput of c blocks per second:

- the throughput of the system is $s \cdot c$ blocks per second,
- the Beacon Chain has a throughput of s blocks per second, and
- each node performs $verification(c) + verification(s) + consensus(n, s)$ computations per second.

If the rate of processing at each node becomes p times, then each node will be able to process $p \cdot (verification(c) + verification(s) + consensus(n, s)) = verification(p * c) + verification(p \cdot s) + consensus(n, p \cdot s)$ computations per second. Thus, the system is able to support $p \cdot s$ number of shards each with a throughput of $p \cdot c$ blocks per second, leading to a total throughput of $(p \cdot c) \cdot (p \cdot s) = p^2 \cdot c \cdot s$ blocks per second. The throughput of the system increases quadratically proportional to the increase in rate of processing of each node.

Security Analysis

There are two relevant security analyses to be made:

- **Consensus Safety**: Safety against the creation of two conflicting finalized chains.
- **Shard Committee Safety**: Safety against an attacker-controlled committee submitting a malicious crosslink.

[6] Each block is assumed to be uniform in the number of operations that its transactions perform.

Consensus safety guarantees are inherited from the Casper FFG, as discussed in Sect. 2.2.

Shard committee safety prevents an attacker from submitting a crosslink for a maliciously created block using a shard committee that it controls. For example, an attacker may create a block that generates ETH from thin air, and this would go unnoticed because only the shard committee actually executes the block to verify its validity. Under a static adversary model,[7] security against these types of attacks is provided by the random sampling of committees. Before proceeding to the security analysis, these are some additional details about the shard committee sampling process:

- At present, there are 64 shards planned.
- A shard committee must be at least 128 validators. If there aren't enough validators to allow for a committee for each shard in every slot, then only some shards will have a committee in a slot.

Now, we can estimate the probability that an attacker controlling some fraction of validators is able to create a crosslink (i.e., control more than two-third of a shard committee). This probability can be derived from the CDF of the hypergeometric distribution, because:

- validators are sampled from the validator set without replacement,
- the attacker controls some fraction of the validator set, and
- the committee is broken if more than two-third of the sample is attacker-controlled

Let's define some notation:

For a random variable X that follows the hypergeometric distribution, let $Pr[X \leq k] = C(N, K, n, k)$ be the hypergeometric cumulative distribution function, where:

- N is the population size,
- K is the number of success states in the population,
- n is the number of draws in each trial, and
- k is the number of observed successes.

In our case, we observe the following:

- X is the number of attacker's validators in a sampled committee,
- N is the validator set size,
- K is the total number of validators under the attacker's control,
- $n = 128$ is the size of a committee,
- $k = \frac{2}{3} \times n$ is the minimum required number of attacker's validator in the sampled committee to break the committee,

[7] A static adversary chooses which validators it controls before the protocol begins. A static adversary cannot, for example, choose to corrupt a validator after looking at the result of the random committee sampling.

- $Pr[X \leq k-1]$ is the probability that the committee is not broken, i.e., the attacker controls less than two-third of the committee,
- $Pr[X \geq k] = 1 - Pr[X \leq k-1]$ is the probability that the committee is broken, i.e., the attacker controls more than two-third of the committee.

So, the probability that a committee is broken is given by

$$Pr[X \geq k] = 1 - C(N, K, 128, \frac{2}{3} \cdot 128).$$

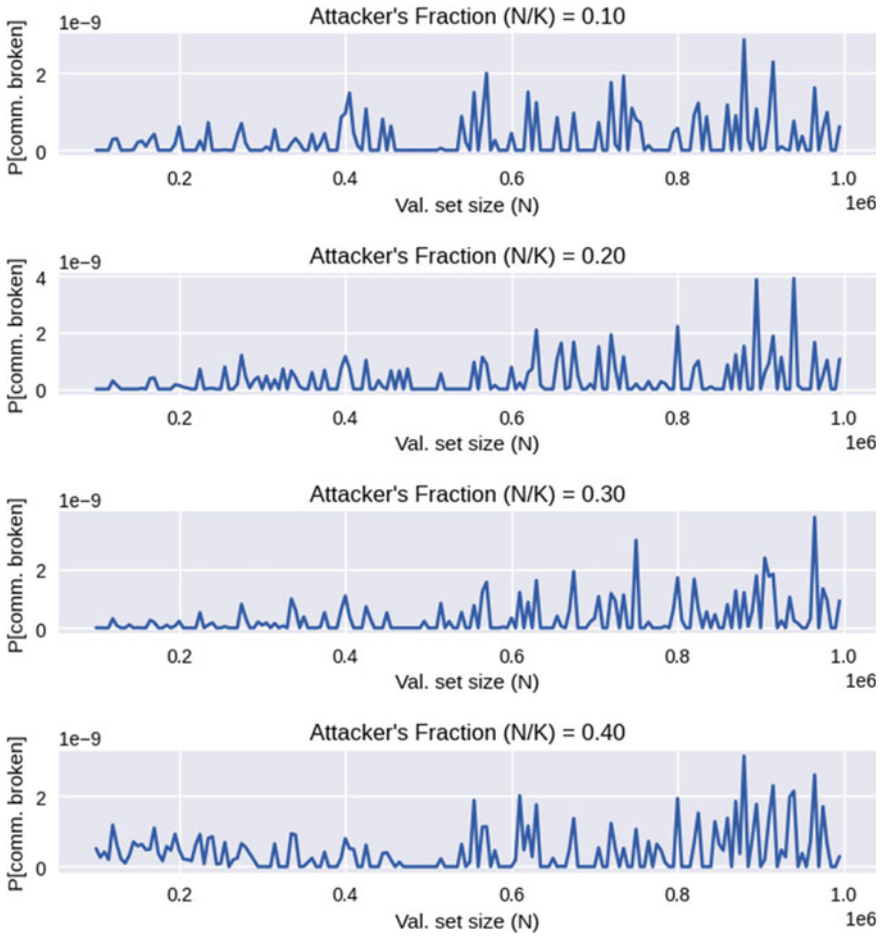


Fig. 4 Probability of a broken shard committee for various fractions of the validator set under the attacker's control

Figure 4 shows the graph of this function for varying values of N and K. The probabilities are lower than $4 \cdot 10^{-9}$, so an attacker is able to execute such an attack every $\frac{10^9}{4}$ slots, which is $\frac{10^9}{4}$ *slots* $\cdot$ 12 *s/slot* $\cdot \frac{1}{365 \cdot 24 \cdot 3600}$ *years/s* $= 95$ *years*.

3 Layer-2 Scaling Solutions

Aggregation-based scaling by relying on the security of L1 for consensus. Layer-2 scaling solutions allow users of a blockchain to execute their transactions on a faster system that operates in parallel to the base blockchain. The transaction system of the layer-2 solution and the base blockchain are able to interact through a smart contract on the base blockchain. Users deposit their assets from the base blockchain into the smart contract associated with the layer-2 system. The side chain maintains a separate state that tracks the ownership of these deposited assets. Users are now

At regular intervals, the state of the layer-2 system is committed to the smart contract on the base blockchain, which is called a checkpoint. It's useful for this state commitment to be in the form of an accumulator[8] with which facts about pieces of the state can be proved, e.g., a Merkle tree root. The smart contract also stores the rules for the layer-2 system's decision-making process. Using these rules, the smart contract ensures that all saved checkpoints are an outcome of the layer-2 system's decision-making process (Fig. 5).

The usual workflow for using a layer-2 scaling solution is as follows:

1. **Deposit**: User deposits their assets from the base blockchain into the smart contract associated with the layer-2 system. When a new asset is deposited in the smart contract, the asset is minted in the layer-2 state and assigned to the corresponding user.
2. **Transact**: Users on the layer-2 system are able to transact with each other using the deposited assets. This only changes the state of the layer-2 system.
3. **Withdraw**: Users wishing to withdraw an asset from the layer-2 system to the base blockchain have to:
    a. burn their asset from the layer-2 state,
    b. commit a checkpoint that contains this updated state to the base blockchain, and
    c. create a transaction on the base blockchain that proves the burning of the asset from the layer-2 state and requests the smart contract to transfer the asset to their address on the base block.

[8] An accumulator is a function that provides information about the membership of an item in a set. In our context, the set is the entire state, and users can check whether a particular piece of the state corresponding to a specific smart contract is indeed included in the current state of the layer-2 system.

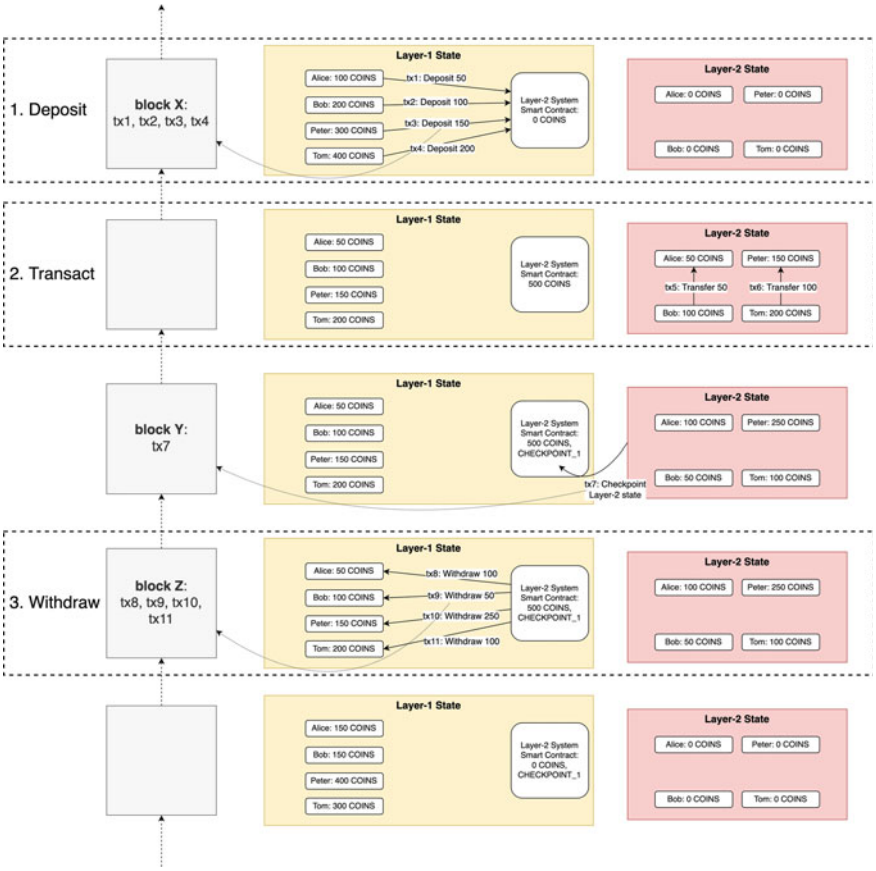


Fig. 5 Users of layer-2 scaling solutions follow the deposit, transact, and withdraw workflow. The scalability comes from aggregating multiple transactions on the layer-2 system (such as `tx5` and `tx6`) into a single checkpoint update transaction on the layer-1 system

3.1 Side Chains

Side chains are layer-2 scaling solutions for which the corresponding smart contract contains the rules of finalization for the chain, but does not contain the rules of its transaction system. The security of a side chain relies on an honest majority assumption about the participant set of its decision-making process.

For example, consider a multi-signature-based system, where any checkpoint update that appears in the smart contract along with a valid multi-signature is considered finalized on the side chain. If the participants of the multi-signature collude to sign on an invalid state transition (such as transferring funds without a valid transaction from the sender) and this update is made in the smart contract, the users have no way of protecting themselves.

A popular side chain on Ethereum is Polygon [7].

3.2 Rollups

Rollups are layer-2 scaling solutions for which the corresponding smart contract contains the rules of its transaction system. There is usually a single designated (but replaceable) actor called the *operator* who submits the checkpoint updates to the smart contract. The operator is expected to verify the validity of the state transition made by the checkpoint update.

There are two types of rollups based on how the state transition rules are verified:

- Optimistic Rollup.
- Zero-Knowledge (ZK) Rollup.

Optimistic Rollup

The rules of finalization of an optimistic rollup include a *challenge period*, which begins after the checkpoint update has been made on the smart contract. During this challenging period, any user can ask for proof of validity of the state transition (such as a valid, signed transaction) in that checkpoint update. When the proof is provided, the smart contract is able to check the validity of the state transition using the rules of the transaction system. If no proof can be produced, the checkpoint update is rejected. At the end of the challenge period, if the checkpoint update has not been rejected, it is deemed finalized.

Two popular optimistic rollups are live at the time of writing: Arbitrum [5] and Optimism [6].

An important distinction between Arbitrum and Optimism is in the way they check the validity of a challenged state transition:

- Optimism uses a single-round challenge [2], where the challenger specifies a previously included transaction to be executed using the Optimism smart contract. The rollup uses EVM transactions, so the rollup's smart contract needs to know to execute EVM code, with appropriate changes to provide the Ethereum chain's context to opcodes that require it. The rollup's smart contract also needs to be able to correctly process the challenge, by executing the challenged transaction on the proper pre-state.[9] All of this is done using an implementation named the Optimistic Virtual Machine.

[9] The transaction defines a state transition, and the rollup's smart contract needs to apply this state transition on the pre-state of the transaction, which is the state of the rollup right before the challenged transaction was included in the rollup.

- Arbitrum uses a multi-round challenge [3], where the challenger and operator (who is defending the transaction they included) communicate back and forth to identify a specific opcode that was executed incorrectly inside a previously executed transaction. The Arbitrum smart contract provides the Arbitrum Virtual Machine, which is able to execute EVM code and support the multi-step challenge process.

Zero-Knowledge (ZK) Rollup

The state transition rules for a ZK Rollup are encoded into a proof system, such that proof of a correctly executed state transition can be made and is computationally cheap to verify.[10] The rules of verification of these proofs are then put into the smart contract for the rollup. Whenever a checkpoint is made on the smart contract, an associated proof is required for the state transition from the last checkpoint. Thus, all checkpoints are automatically verified to be the result of correctly executed state transition, without the need for challenge periods that appear to be optimistic rollups.

A number of ZK Rollups exist at the time of writing: Loopring [4], StarkNet [8], and zkSync [9].

4 Conclusion

Blockchain technology is a vast domain of computer science that is yet to be fully explored. Given the current levels of interest and resources being deployed into blockchain research, this space will undoubtedly see rapid development in the coming years. With the increasing adoption of Web3 in mainstream industries, blockchain scalability has become a crucial research area with immediate consequences. Over the course of this chapter, we've explored the considerations in scalability solutions, the general categories of scaling solutions, and then a further deeper exploration of the leading solutions. While this chapter aims to serve as a gentle introduction to the topic of blockchain scaling, the context provided through this chapter will also enable readers to understand and analyze other scaling solutions and related technologies.

References

1. Ethereum virtual machine (evm) | ethereum.org. https://ethereum.org/en/developers/docs/evm/. Accessed 02 Jan 2022

[10] Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (ZK-SNARKS) are a popular choice for the proof system in ZK-Rollups. Although the zero-knowledge properties are not essential to this type of rollup, the name has stuck.

2. How does optimism's rollup really work? | paradigm research. https://research.paradigm.xyz/optimism. Accessed 31 Dec 2021
3. Inside arbitrum - offchain labs dev center. https://developer.offchainlabs.com/docs/inside_arbitrum. Accessed 31 Dec 2021
4. Loopring - zkrollup exchange and payment protocol. https://loopring.org. Accessed 22 Sept 2021
5. Offchain labs. https://offchainlabs.com/. Accessed 22 Sept 2021
6. Optimism. https://optimism.io/. Accessed 22 Sep 2021
7. Polygon | ethereum's internet of blockchains. https://polygon.technology/. Accessed 22 Sep 2021
8. Starknet - starkware industries ltd. https://starkware.co/product/starknet/. Accessed 22 Sept 2021
9. zksync - rely on math, not validators. https://zksync.io/. Accessed 22 Sept 2021
10. Buterin, V., Griffith, V.: Casper the friendly finality gadget (2017). CoRR, arXiv:abs/1710.09437
11. Neu, J., Tas, E.N., Tse, D.: The availability-accountability dilemma and its resolution via accountability gadgets (2021). CoRR, arXiv:abs/2105.06075