ZK-rollup 调研

核心理念是将大量的计算和状态迁移的链下，链上存放部分的状态，接收链下的状态变更，并验证链下产生的状态变更。

Rollup 中又有两种实现方案，分别是 Optimistic Rollup 和 zk-Rollup 。其中：

- Optimistic Rollup 采用欺诈证明（Fraud proof），首先假设上传到主链所有交易都是合法的，并设置挑战期，允许验证者提出证明来挑战有问题的交易，一旦证实有欺诈行为发生，就对欺诈者进行惩罚，对挑战者进行奖励。
- Zk-Rollup 则采用有效性证明（Validity proof），在链下进行所有交易的验证和打包，验证后的交易被提交到主链时附上零知识证明来证明交易的有效性。

Zero-knowledge rollups (ZK-rollups) are layer 2 scaling solutions that increase throughput on Ethereum Mainnet by moving computation and state-storage off-chain. ZK-rollups can process thousands of transactions in a batch and then only post some minimal summary data to Mainnet.

zk Rollup 的本质是将链上的用户状态压缩存储在一棵Merkle树中，并将用户状态的变更转移到链下来，同时通过 zkSNARK 的证明来保证该链下用户状态变更过程的正确性。在链上直接处理用户状态的变更成本是比较高的，但是仅仅利用链上的智能合约来验证一个零知识证明的 PROOF 是否正确，成本是相对低很多的。另外必要的转账信息也会被和证明一起提交到合约，方便用户查账。

https://trapdoortech.medium.com/l2-deep-into-zksync-source-code-d6be1b3c16e5

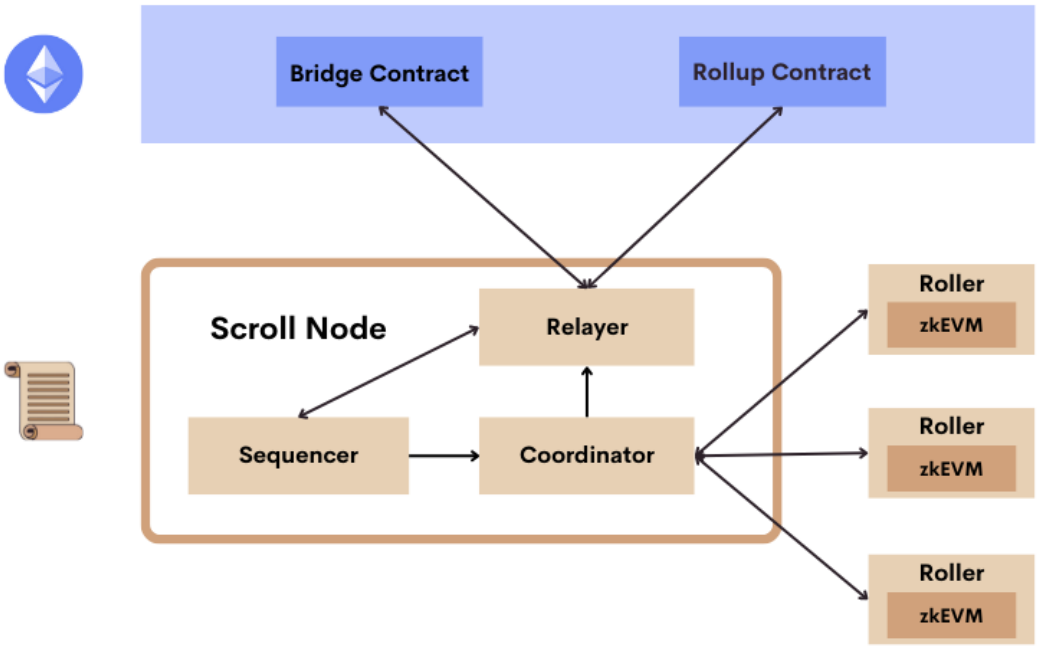
Rollup 中主要有三个角色：

- Sequencer：主要负责从用户那里收集交易，对它们排序并将新的 Merkle 根传到 Layer1 的 rollup 智能合约上，类似于矿工。
- Prover：负责计算验证所有的交易，并生成一个 zk-proof 来证明交易的有效性。
- Verifier：通常被部署在 Layer 1 上，负责完成部分计算来验证 Prover 提交的 proof 的有效性，以确保其提供了计算诚实性所需的所有的信息。

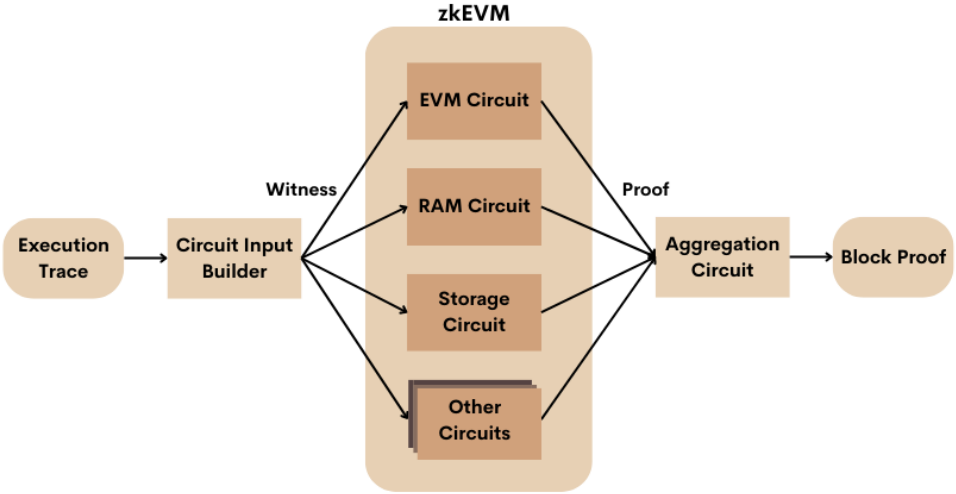
Scroll

The current architecture consists of three infrastructure components (see Figure 1):

- **Scroll Node:** Constructs L2 blocks from user transactions, commits them to the Ethereum base layer, and passes messages between L1 and L2.
- **Roller Network:** Generates the zkEVM validity proofs to prove that transactions are executed correctly.
- **Rollup and Bridge Contracts:** Provides data availability for Scroll transactions, verifies zkEVM validity proofs, and allows users to move assets between Ethereum and Scroll



- Sequencer 通过rpc 获取 L2的交易集合、打包、执行生成新的L2区块和状态根，该协议的 sequencer 基于以太坊客户端实现。当出现新的区块后， coordinator从sequence处获取区块执行产生的trace，并将trace随机分发给roller 进行proof的生成。 Relayer 负责监视部署在 Ethereum和Scroll 上 Rollup contract 和 Bridge Contract 以跟踪L2 block的状态 （数据可用性和有效性证明）以及 Bridge Contract 中产生的充值和取钱的事件，进行信息传递。
- roller 是网络中的证明器，负责生成proof。Scroll 采用了GPU ASICS等减少证明事件和证明消耗。



- Rollup Contract 接收来自 L2的状态根信息和区块信息（区块信息不存放在state中，所以其也不会造成大量的gas fee）该合约负责验证proof的有效性，验证通过后，该区块的被最终确定
- Bridge Contract 负责在L1 与 L2 之间传递任意的消息，以及支持 ERC20 token 的转移。 ETH->SCROLL，用户首先调用 sendMessage 函数，Relayer 在监听到该笔交易后，将交易传递给 sequencer 等待在L2上的处理。

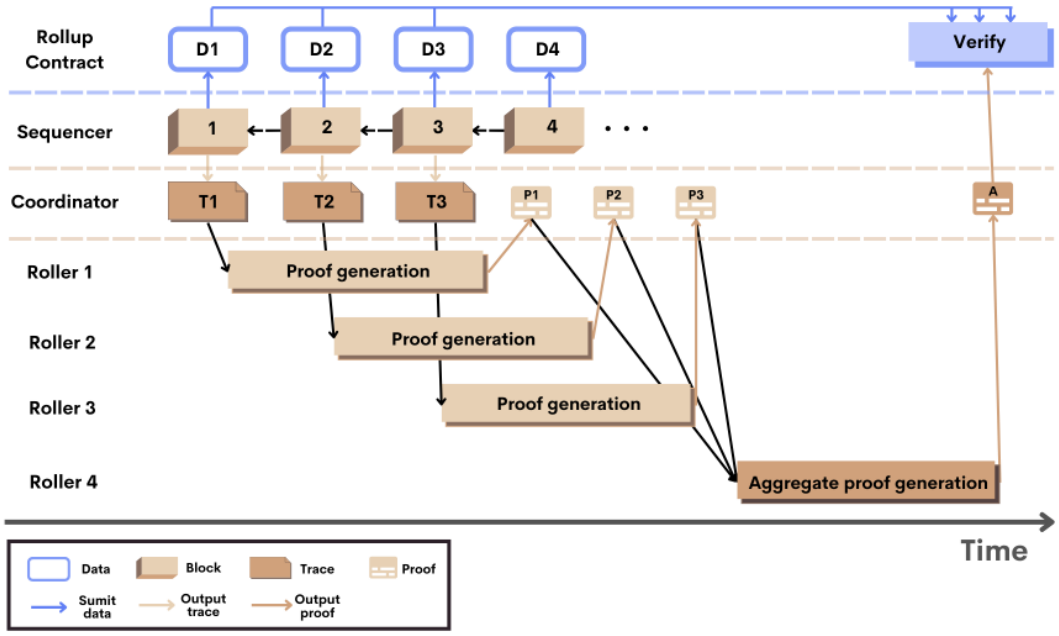


Figure 3. Scroll workflow

L2 blocks in Scroll are generated, committed to base layer Ethereum, and finalized in the following sequence of steps:

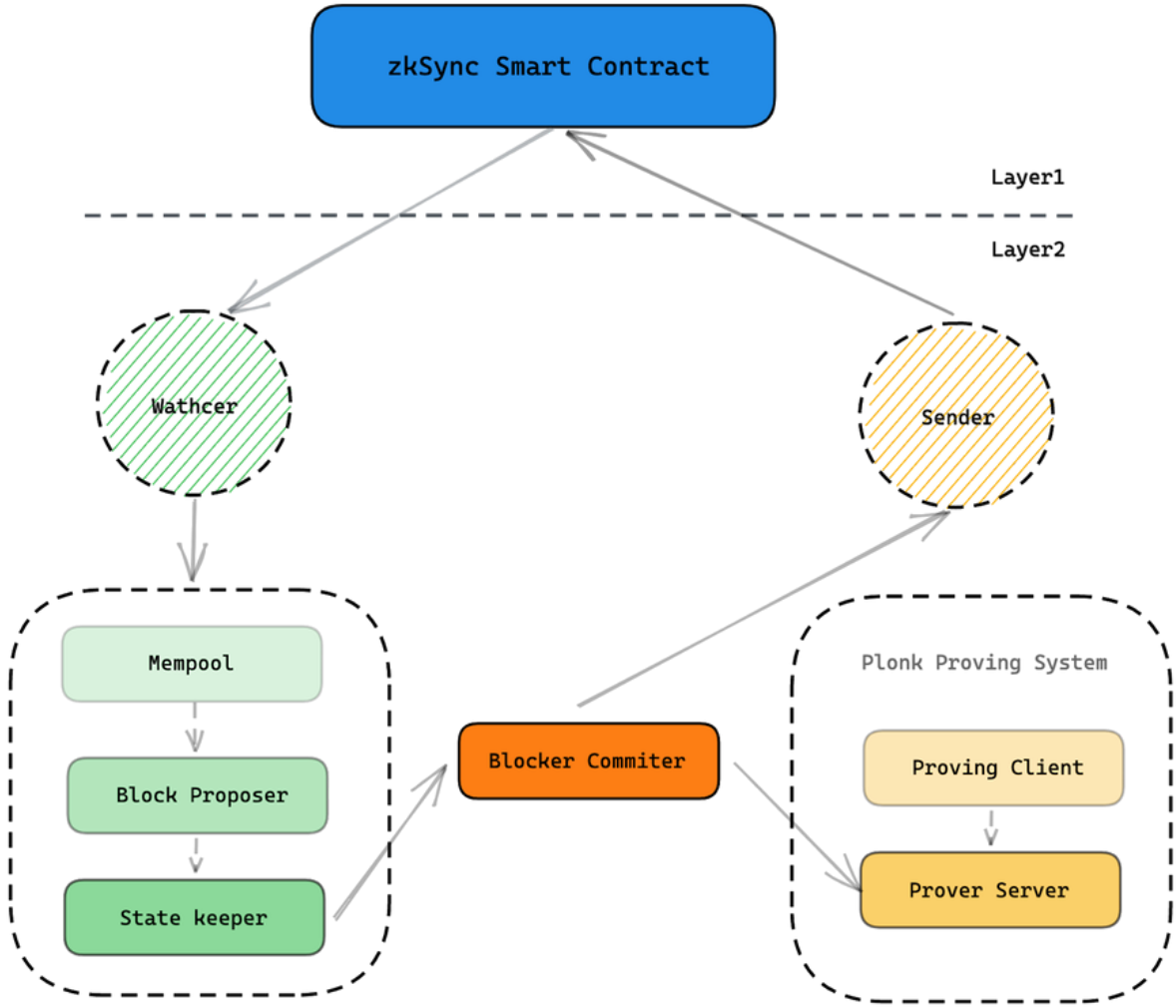
1. The Sequencer generates a sequence of blocks. For the *i*-th block, the Sequencer generates an execution trace ***T*** and sends it to the Coordinator. Meanwhile, it also submits the transaction data ***D*** as calldata to the Rollup contract on Ethereum for data availability and the resulting state roots and commitments to the transaction data to the Rollup contract as state.
2. The Coordinator randomly selects a Roller to generate a validity proof for each block trace. To speed up the proof generation process, proofs for different blocks can be generated in parallel on different Rollers.
3. After generating the block proof ***P*** for the *i*-th block, the Roller sends it back to the Coordinator. Every *k* blocks, the Coordinator dispatches an aggregation task to another Roller to aggregate *k* block proofs into a single aggregate proof ***A***.
4. Finally, the Coordinator submits the aggregate proof ***A*** to the Rollup contract to finalize L2 blocks *i+1* to *i+k* by verifying the aggregate proof against the state roots and transaction data commitments previously submitted to the rollup contract.

Figure 3 illustrates that Scroll blocks will be finalized on L1 in a multi-step process. Each L2 block will progress through the following three stages until it is finalized.

- `Precommitted` indicates a block has been proposed by a Sequencer and sent to Rollers. Although Precommitted blocks are not yet a canonical part of the Scroll L2 chain because they have not been posted on the Ethereum base layer, users who trust the Sequencer can choose to take action on them in anticipation.
- `Committed` indicates the transaction data of this block has been posted on the rollup contract on Ethereum. This ensures that the block data is available, but does not prove that it has been executed in a valid way.
- `Finalized` indicates the correct execution of transactoins in this block has been proven by verifying a validity proof on-chain on Ethereum. Finalized blocks are considered canonical parts of the Scroll L2 chain.

zkSync

支持 ETH 和 ERC20 转账。 用户签署交易并提交给验证人； 验证人将数千笔交易 Rollup 到一个区块中，并向主网的智能合约提交新状态的默克尔根和加密证明（SNARK），证明新状态是基于旧状态的正确更新；除了证明以外，每笔交易的数据都会使用廉价的 Calldata 在主网发布，以便任何人在任何时刻都可以重建状态；证明和状态均由智能合约验证，从而验证区块中所有交易的有效性和区块数据的可用性。



主要架构分为链上和链下，L1 的核心为智能合约，主要负责存款、提款、交易验证；L2 分为 L1 交互（Watcher、Sender）、L2 状态维护（Mempool、Block Proposer、State Keeper、Block Commiter）、零知识证明系统。

当用户想要存款时，调用 L1 zkSync 智能合约存储资金；Watcher 监控 L1 存款交易，当交易发生时则会放入 Mempool 中；Block Proposer 处理 Mempool 交易打包，并提交 State Keeper 更新账本。

当用户想要使用 L2 低成本快速转账时，调用 zkSync API 提交转账交易；交易同样会按照流程流转至 Mempool > Block Proposer > State Keeper；最终 State Keeper 通知 Block Commiter 收集生成零知识证明所需信息，调用 Plonk Proving System 生成零知识证明后，借助 Sender 将存款和转账等交易数据，以及将对应的零知识证明提交到 L1 的 zkSync 智能合约验证；等待 L1 交易确认后，Watcher 会通知 L2 更新交易状态为最终确认。

核心理念： 大多数的数据存储在链下、交易在链下执行，但是所有链下交易的正确性都是可以在链上证明的，所以与链上交易的安全等级相同。

- 用户之间可以进行资产的转移、充值
- 用户可以将资产提取到layer1 所在的账户

Rollup operation requires the assistance of an operator, who rolls transactions together, computes a zero-knowledge proof of the correct state transition, and affects the state transition by interacting with the rollup contract

所有的Rollup方案都引入一个链下方，其负责执行这些交易，并调用rollup的智能合约将状态更新验证、上链

Rollup operation requires the assistance of an operator, who rolls transactions together, computes a zero-knowledge proof of the correct state transition, and affects the state transition by interacting with the rollup contract. To understand the design, we need to look into how zkSync rollup transactions work.

zkSync operations are divided into rollup transactions (initiated inside rollup by a rollup account) and priority operations (initiated on the mainchain by an Ethereum account).

The zkSync rollup operation lifecycles are as follows:

- A user creates a transaction or a priority operation.
- After processing this request, the operator creates a rollup operation and adds it to the block.
- Once the block is complete, the operator submits it to the zkSync smart contract as a block commitment. Part of the logic of some rollup operations is checked by the smart contract.
- The proof for the block is submitted to the zkSync smart contract as block verification. If the verification succeeds, the new state is considered final.

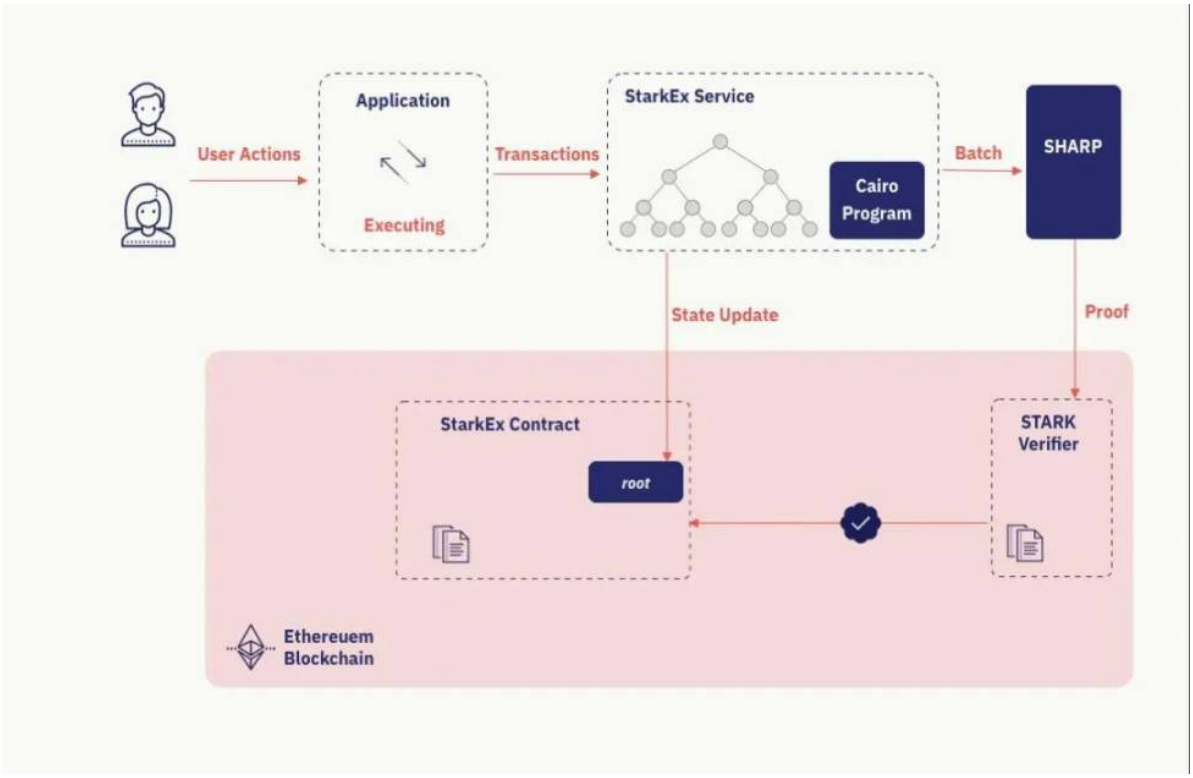
Furthermore, on zkSync, each L2 block will progress through the following four stages until it is final.

- `Pending`: The transaction was received by the operator, but it has not been processed yet.
- `Processed`: The transaction is processed by the operator and is confirmed to be included in the next block.
- `Committed`: This indicates that the transaction data of this block has been posted on Ethereum. It does not prove that it has been executed in a valid way, but it ensures the availability of the block data.
- `Finalized`: This indicates that the SNARK validity proof for the transaction has been submitted and verified by the smart contract. After this step, the transaction is considered to be final.

The typical time for a transaction to go from `Processed` to `Finalized` is a couple of hours at the current stage.

Please note that for developer convenience, we usually treat the `Processed` and `Committed` states as a single stage called `Committed` since they have no difference from the UX/DexEx standpoints.

Starkware



1. 用户首先在应用中进行操作，通常是应用的客户端或者网页前端，应用执行这些动作后向 StarkEx 服务发送交易；
2. StarkEx 服务把这些交易打包执行后把包裹发送给 SHARP（Shared prover，共享的证明服务），这个过程中的执行是由 Cairo 语言编写的程序完成的，值得一提的是，StarkWare 通过 Cairo 这一图灵完备的语言，将所有智能合约的计算结果转化成了可证明的多项式方程，由此让智能合约与有效性证明兼容；
3. SHARP 是一个基于 Stark 的证明系统 ，由它来生成能证明这个批次交易有效性的证明；
4. SHARP 把 Stark 证明发给部署在 Layer 1 上的 Verifier，由 Verifier 来完成验证；
5. StarkEx Service 会发送一个链上状态更新交易给 Layer 1 上部署的 StarkNet 智能合约，这个智能合约只会在 Verifier 完成了证明之后才会接受这个新的状态变更。