Trustless log index

~~~~~~~~~~~~~~

Where core values meet scalability

EthBoulder 2026

Zsolt Felföldi

Ethereum Foundation

Geth client developer

zsfelfoldi@ethereum.org
~~~~~~~~~~~~~~

Logs vs State

State	Logs (now)	Logs (with TLI)
Mutable key => value store	Append-only, content/time addressed storage	Append-only, content/time addressed storage
Read (key lookup): entire local state or witnesses; O(log(N)) cost	Read (value/pattern search): entire chain (extremely expensive); O(N) cost	Read (value/pattern search): index table lookups; **O(log(N)^2) cost**
Write: entire local state or witnesses; O(log(N)) cost	Write: no history required; O(1) cost	Write: last 80 blocks required; O(log(N)) **mostly async/in-memory;** **cca 50x faster than state tree**
Huge, constantly changing dataset (hard to sync, high entry barrier, infrastructural capture risk)	Easy to sync, finalized block receipts are not changing	Easy to sync, block receipts and finalized index tables are not changing
Expire/resurrect is hard; cannot be done in a backwards compatible way	Easy to expire/resurrect when needed	Easy to expire/resurrect when needed
Suitable for sequential execution	Suitable for parallel execution (no collisions)	Suitable for parallel execution (no collisions)

Index tables

- An ordered list of events associated with a certain block range, searchable by event type and index value
  - Block hashes
  - Transaction hashes
  - Log addresses
  - Log topics
- Tree-hashed in SSZ List format, allows inclusion or exclusion proofs of certain events or combinations of them in its block range, with efficient Merkle proofs
- Index tables are generated once the entire block range exists
  - Existing tables covering adjacent block ranges can be merged into bigger tables
- Chain history can be covered with multiple index tables
  - Single-block tables generated for each new head block
  - Smaller tables merged into bigger ones when possible
  - Bigger tables covering older history allow more efficient proofs

Index entries (example)

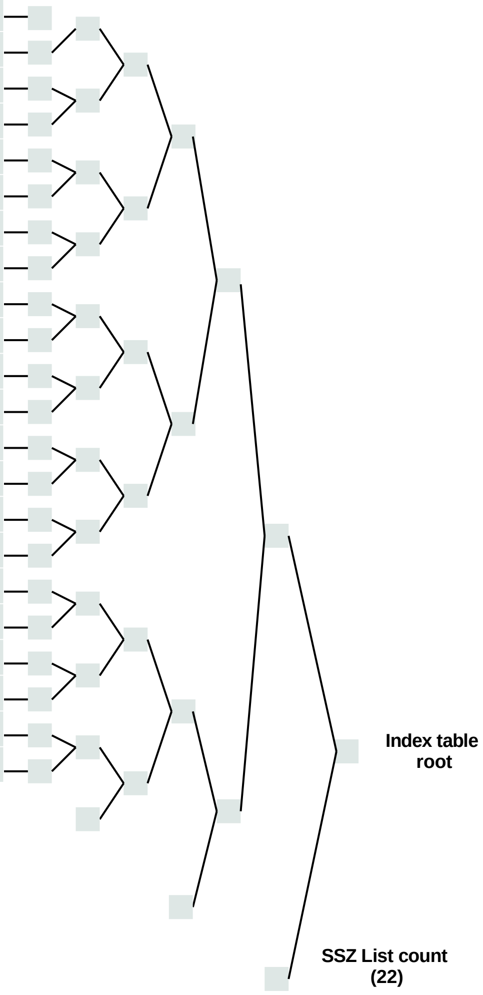
- Block #40: empty
- Block #41: empty
- Block #42:
  - Tx #0:
    - WETH Transfer from Alice to Bob
    - USDT Transfer from Bob to Alice
  - Tx #1: WETH Withdrawal by Bob
- Block #43:
  - Tx #0: USDT Transfer from Alice to Carol

Block #	Tx #	Log #	Type	Value
40			0 (Block)	0xbf98e6cb26f6ff... (block hash)
41			0 (Block)	0x42f66a2e9f9c68... (block hash)
42	0		1 (Tx)	0xca2d12d1b8132d... (tx hash)
42	0	0	2 (Address)	0x00..00c02aaa39... (WETH)
42	0	0	3 (Topic0)	0xddf252ad1be2c8... (Transfer)
42	0	0	4 (Topic1)	0x00..004a5e9a3b... (Alice)
42	0	0	5 (Topic2)	0x00..00f2a5fd4d... (Bob)
42	0	1	2 (Address)	0x00..00dac17f95... (USDT)
42	0	1	3 (Topic0)	0xddf252ad1be2c8... (Transfer)
42	0	1	4 (Topic1)	0x00..00f2a5fd4d... (Bob)
42	0	1	5 (Topic2)	0x00..004a5e9a3b... (Alice)
42	1		1 (Tx)	0xa75590c9ced728... (tx hash)
42	1	0	2 (Address)	0x00..00c02aaa39... (WETH)
42	1	0	3 (Topic0)	0x7fcf532c15f0a6... (Withdrawal)
42	1	0	4 (Topic1)	0x00..00f2a5fd4d... (Bob)
42			0 (Block)	0x66f42ef12b140e... (block hash)
43	0		1 (Tx)	0x7046035b326ab2... (tx hash)
43	0	0	2 (Address)	0x00..00dac17f95... (USDT)
43	0	0	3 (Topic0)	0xddf252ad1be2c8... (Transfer)
43	0	0	4 (Topic1)	0x00..004a5e9a3b... (Alice)
43	0	0	5 (Topic2)	0x00..00ac8cbe20... (Carol)
43			0 (Block)	0x978ce0036b6d1c... (block hash)

Sorting index entries

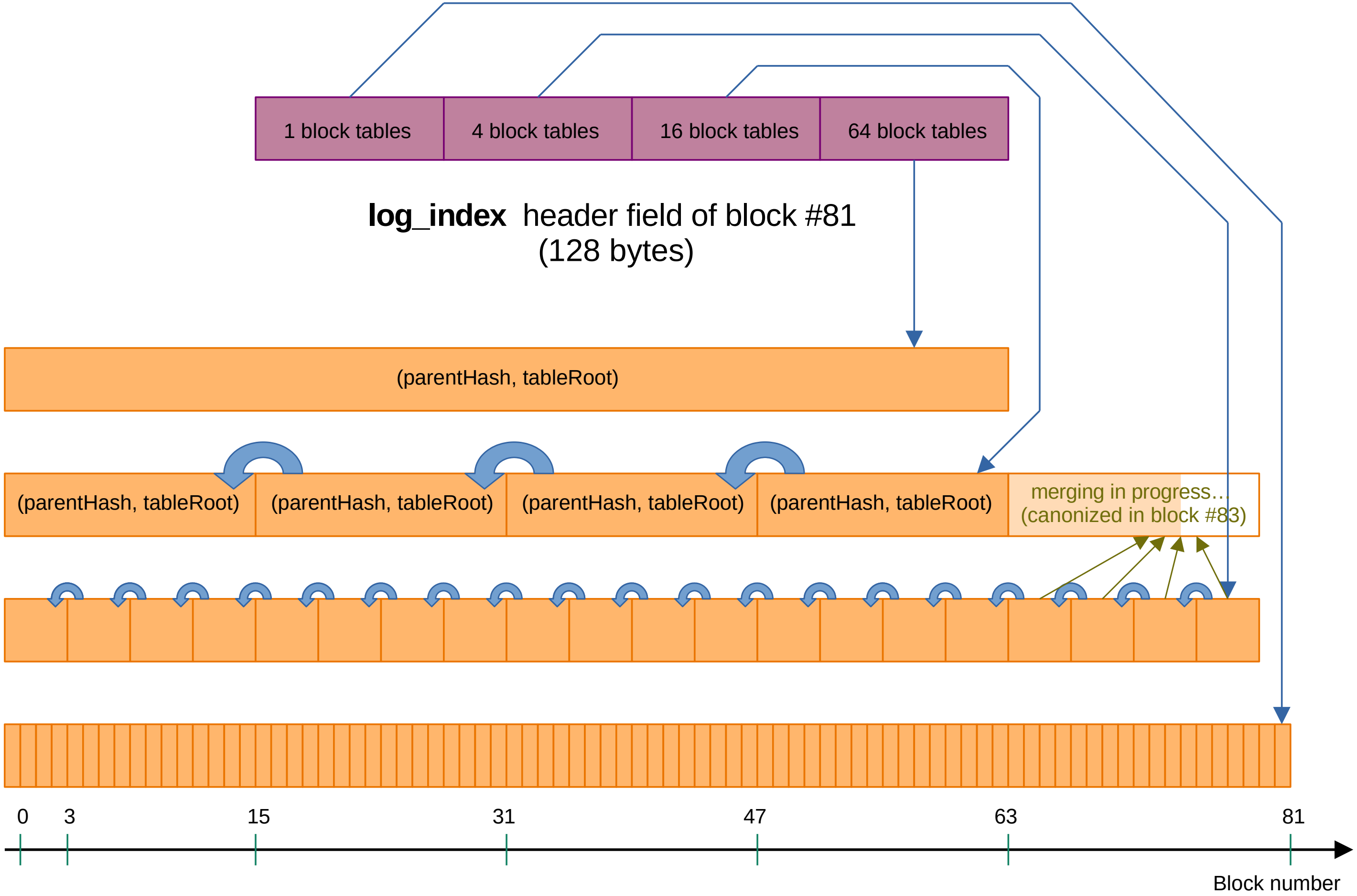
- Each entry is encoded for hashing as 64 bytes (storage/network encoding can be more compact)
  - Event type (8 bytes, big endian)
  - Index value (32 bytes)
  - Block number (8 bytes, big endian)
  - Transaction index (8 bytes, big endian)
  - Log index (8 bytes, big endian)
- Transaction/log index is zero where not applicable
- Entries are lexicographically ordered according to the hashed encoding

Type	Value	Block #	Tx #	Log #
0 (Block)	0x42f66a2e9f9c68... (block hash)	41		
0 (Block)	0x66f42ef12b140e... (block hash)	42		
0 (Block)	0x978ce0036b6d1c... (block hash)	43		
0 (Block)	0xbf98e6cb26f6ff... (block hash)	40		
1 (Tx)	0x7046035b326ab2... (tx hash)	43	0	
1 (Tx)	0xa75590c9ced728... (tx hash)	42	1	
1 (Tx)	0xca2d12d1b8132d... (tx hash)	42	0	
2 (Address)	0x00..00c02aaa39... (WETH)	42	0	0
2 (Address)	0x00..00c02aaa39... (WETH)	42	1	0
2 (Address)	0x00..00dac17f95... (USDT)	42	0	1
2 (Address)	0x00..00dac17f95... (USDT)	43	0	0
3 (Topic0)	0x7fcf532c15f0a6... (Withdrawal)	42	1	0
3 (Topic0)	0xddf252ad1be2c8... (Transfer)	42	0	0
3 (Topic0)	0xddf252ad1be2c8... (Transfer)	42	0	1
3 (Topic0)	0xddf252ad1be2c8... (Transfer)	43	0	0
4 (Topic1)	0x00..004a5e9a3b... (Alice)	42	0	0
4 (Topic1)	0x00..004a5e9a3b... (Alice)	43	0	0
4 (Topic1)	0x00..00f2a5fd4d... (Bob)	42	0	1
4 (Topic1)	0x00..00f2a5fd4d... (Bob)	42	1	0
5 (Topic2)	0x00..004a5e9a3b... (Alice)	42	0	1
5 (Topic2)	0x00..00ac8cbe20... (Carol)	43	0	0
5 (Topic2)	0x00..00f2a5fd4d... (Bob)	42	0	0



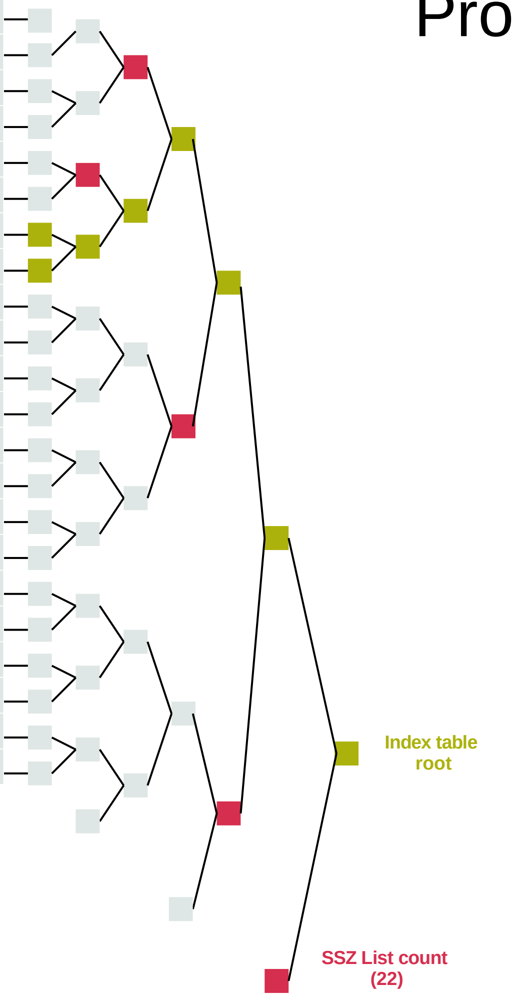
▨ Tree node

Covering the chain history with table chains



Proof example: exclusion proof

Type	Value	Block #	Tx #	Log #
0 (Block)	0x42f66a2e9f9c68... (block hash)	41		
0 (Block)	0x66f42ef12b140e... (block hash)	42		
0 (Block)	0x978ce0036b6d1c... (block hash)	43		
0 (Block)	0xbf98e6cb26f6ff... (block hash)	40		
1 (Tx)	0x7046035b326ab2... (tx hash)	43	0	
1 (Tx)	0xa75590c9ced728... (tx hash)	42	1	
1 (Tx)	0xca2d12d1b8132d... (tx hash)	42	0	
2 (Address)	0x00..00c02aaa39... (WETH)	42	0	0
2 (Address)	0x00..00c02aaa39... (WETH)	42	1	0
2 (Address)	0x00..00dac17f95... (USDT)	42	0	1
2 (Address)	0x00..00dac17f95... (USDT)	43	0	0
3 (Topic0)	0x7fcf532c15f0a6... (Withdrawal)	42	1	0
3 (Topic0)	0xddf252ad1be2c8... (Transfer)	42	0	0
3 (Topic0)	0xddf252ad1be2c8... (Transfer)	42	0	1
3 (Topic0)	0xddf252ad1be2c8... (Transfer)	43	0	0
4 (Topic1)	0x00..004a5e9a3b... (Alice)	42	0	0
4 (Topic1)	0x00..004a5e9a3b... (Alice)	43	0	0
4 (Topic1)	0x00..00f2a5fd4d... (Bob)	42	0	1
4 (Topic1)	0x00..00f2a5fd4d... (Bob)	42	1	0
5 (Topic2)	0x00..004a5e9a3b... (Alice)	42	0	1
5 (Topic2)	0x00..00ac8cbe20... (Carol)	43	0	0
5 (Topic2)	0x00..00f2a5fd4d... (Bob)	42	0	0



⬜ Tree node (not relevant for the proof)

🟥 Proof node (included in the proof)

🟨 Hash recalculated by verifier

Block range: #40..#43

- event type: 1 (transaction hash)

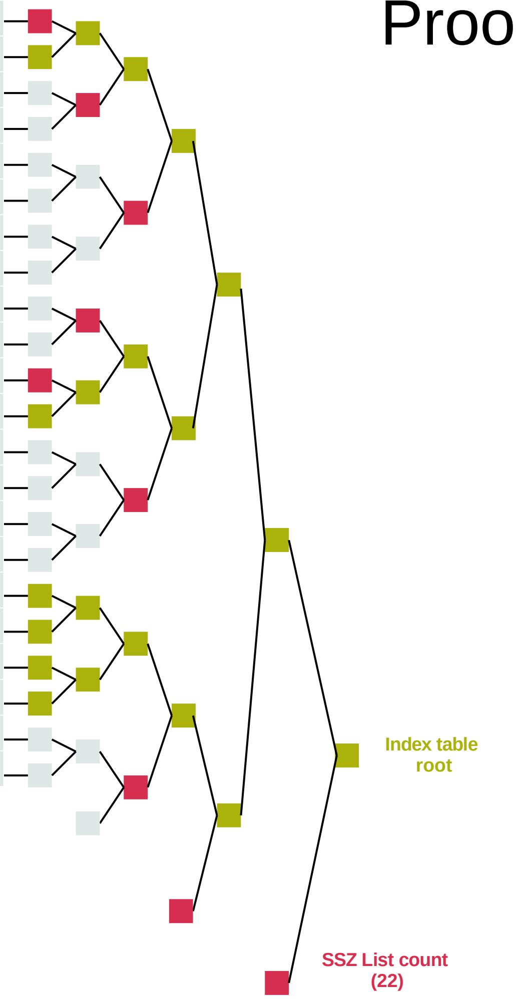
- index value: 0xdeadbeefcafe...

Results: not found

- Prover looks up the searched entry in the table using binary search
- Searched entry does not exist, finds two adjacent entries
- Creates a multi-proof proving those two entries
- Proof cost per index table: two entries (cca 40-45 bytes per entry with compact encoding) plus *ceil(log2(count))* 32 byte proof nodes

Proof example: log pattern search

Type	Value	Block #	Tx #	Log #
0 (Block)	0x42f66a2e9f9c68... (block hash)	41		
0 (Block)	0x66f42ef12b140e... (block hash)	42		
0 (Block)	0x978ce0036b6d1c... (block hash)	43		
0 (Block)	0xbf98e6cb26f6ff... (block hash)	40		
1 (Tx)	0x7046035b326ab2... (tx hash)	43	0	
1 (Tx)	0xa75590c9ced728... (tx hash)	42	1	
1 (Tx)	0xca2d12d1b8132d... (tx hash)	42	0	
2 (Address)	0x00..00c02aaa39... (WETH)	42	0	0
2 (Address)	0x00..00c02aaa39... (WETH)	42	1	0
2 (Address)	0x00..00dac17f95... (USDT)	42	0	1
2 (Address)	0x00..00dac17f95... (USDT)	43	0	0
3 (Topic0)	0x7fcf532c15f0a6... (Withdrawal)	42	1	0
3 (Topic0)	0xddf252ad1be2c8... (Transfer)	42	0	0
3 (Topic0)	0xddf252ad1be2c8... (Transfer)	42	0	1
3 (Topic0)	0xddf252ad1be2c8... (Transfer)	43	0	0
4 (Topic1)	0x00..004a5e9a3b... (Alice)	42	0	0
4 (Topic1)	0x00..004a5e9a3b... (Alice)	43	0	0
4 (Topic1)	0x00..00f2a5fd4d... (Bob)	42	0	1
4 (Topic1)	0x00..00f2a5fd4d... (Bob)	42	1	0
5 (Topic2)	0x00..004a5e9a3b... (Alice)	42	0	1
5 (Topic2)	0x00..00ac8cbe20... (Carol)	43	0	0
5 (Topic2)	0x00..00f2a5fd4d... (Bob)	42	0	0



□ Tree node (not relevant for the proof)

🟥 Proof node (included in the proof)

🟩 Hash recalculated by verifier

Block range: #40..#43

Address: 0x00..00dac17f95... (USDT)

Topic0: 0xddf252ad1be2c8... (Transfer)

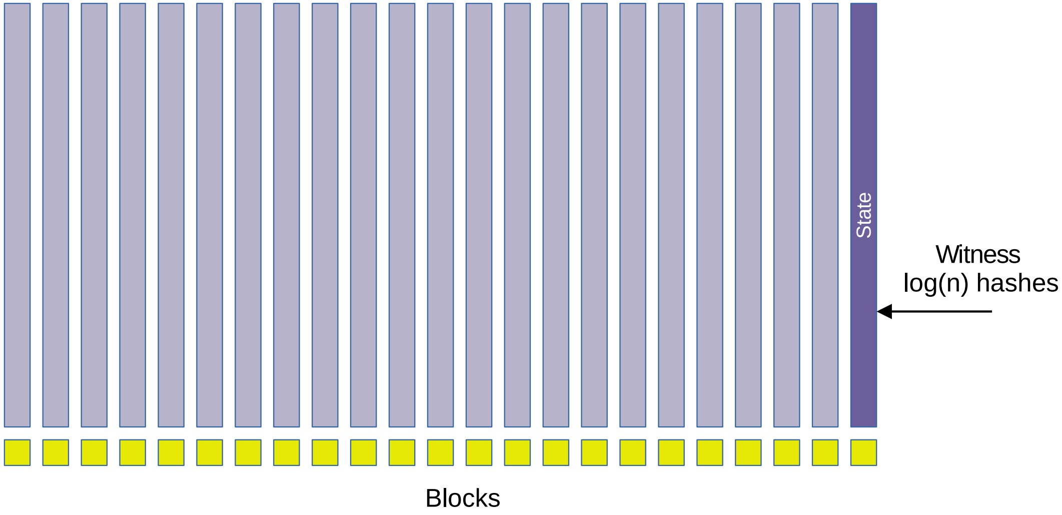
Topic1: 0x00..00f2a5fd4d... (Bob)

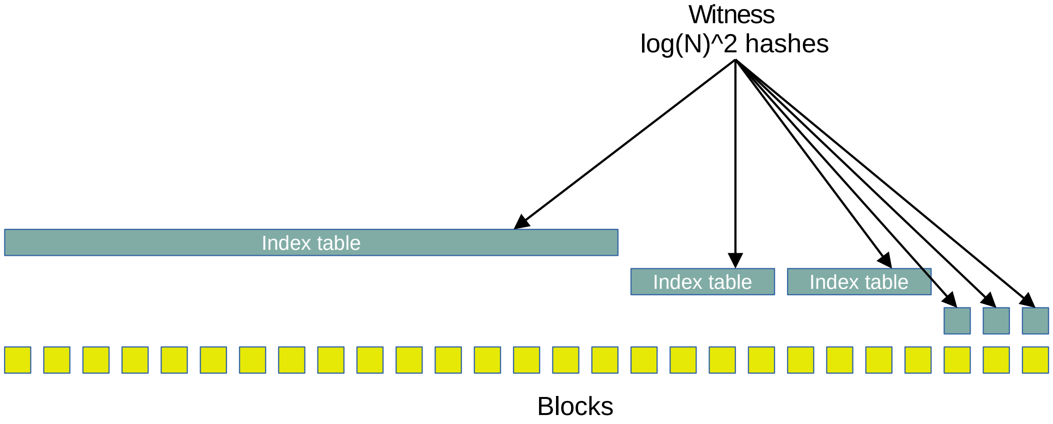
Topic2: anything

Results: [{*Block #42 Tx #0 Log #1*}]

- All occurences of **Topic1 = Bob** are proven:
  - *{Block #42 Tx #0 Log #1}*
  - *{Block #42 Tx #1 Log #0}*
- **Topic0 = Withdrawal** in *{Block #42 Tx #1 Log #0} (no match)*
- *Block #42* root hash is proven
  - Receipt proves that *{Block #42 Tx #0 Log #1} is a match*

	Log index in protocol	Indexing with ZK proofs
Last few hours	• Limited size tables can be safely processed by every block builder/validator • Proof size efficiency is sufficient (last hour is 10-15 tables, every extra hour 5 tables; proof overhead per table is around 400-700 bytes) • Suitable for off-chain/cross-chain event proofs	• Proofs are required frequently and with low latency (unindexed search costs 15-20 Mb per hour of chain history) • Still, the most valuable information (most recent events) are hard for regular users to access trustlessly • High costs, information asymmetry • Unsuitable for off-chain/cross-chain event proofs
Long history	• With limited size tables, proof size efficiency for long history lookups is insufficient • Processing very large index structures in consensus is risky for the whole protocol	• Big index tables can be processed with hours or days of latency • Proving efforts can be shared between many parties • Low cost • Very large tables can be safely created (highest possible proof size efficiency for long history lookups)





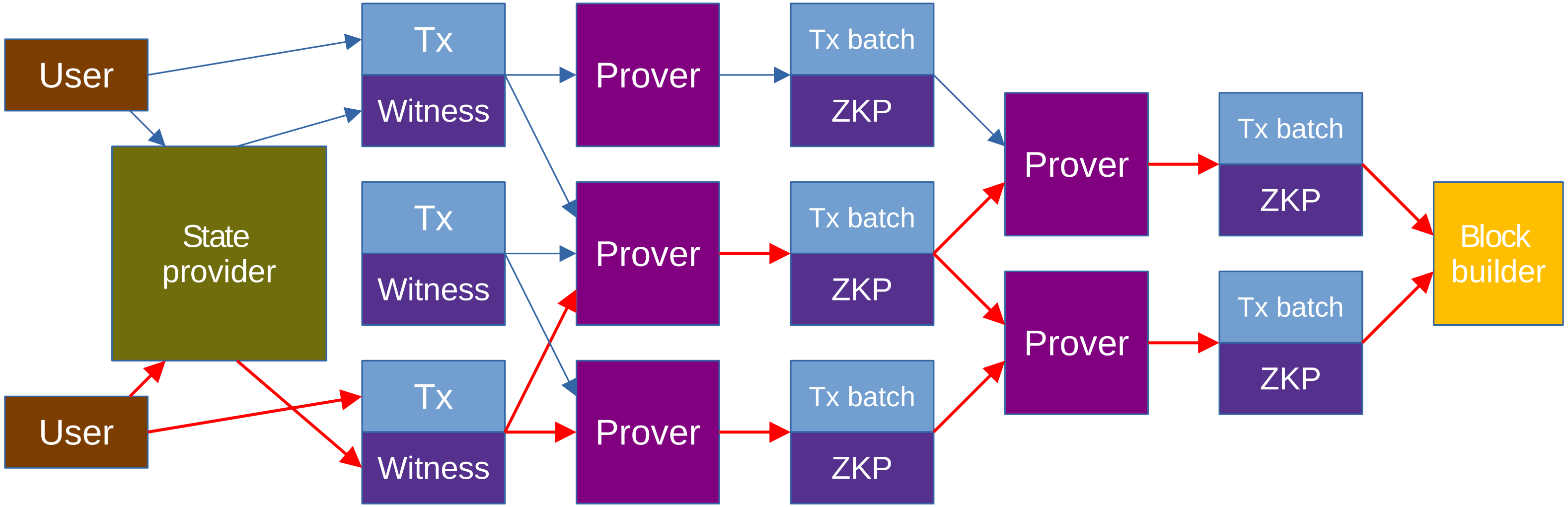
Pro state tree:

- Smaller witness size

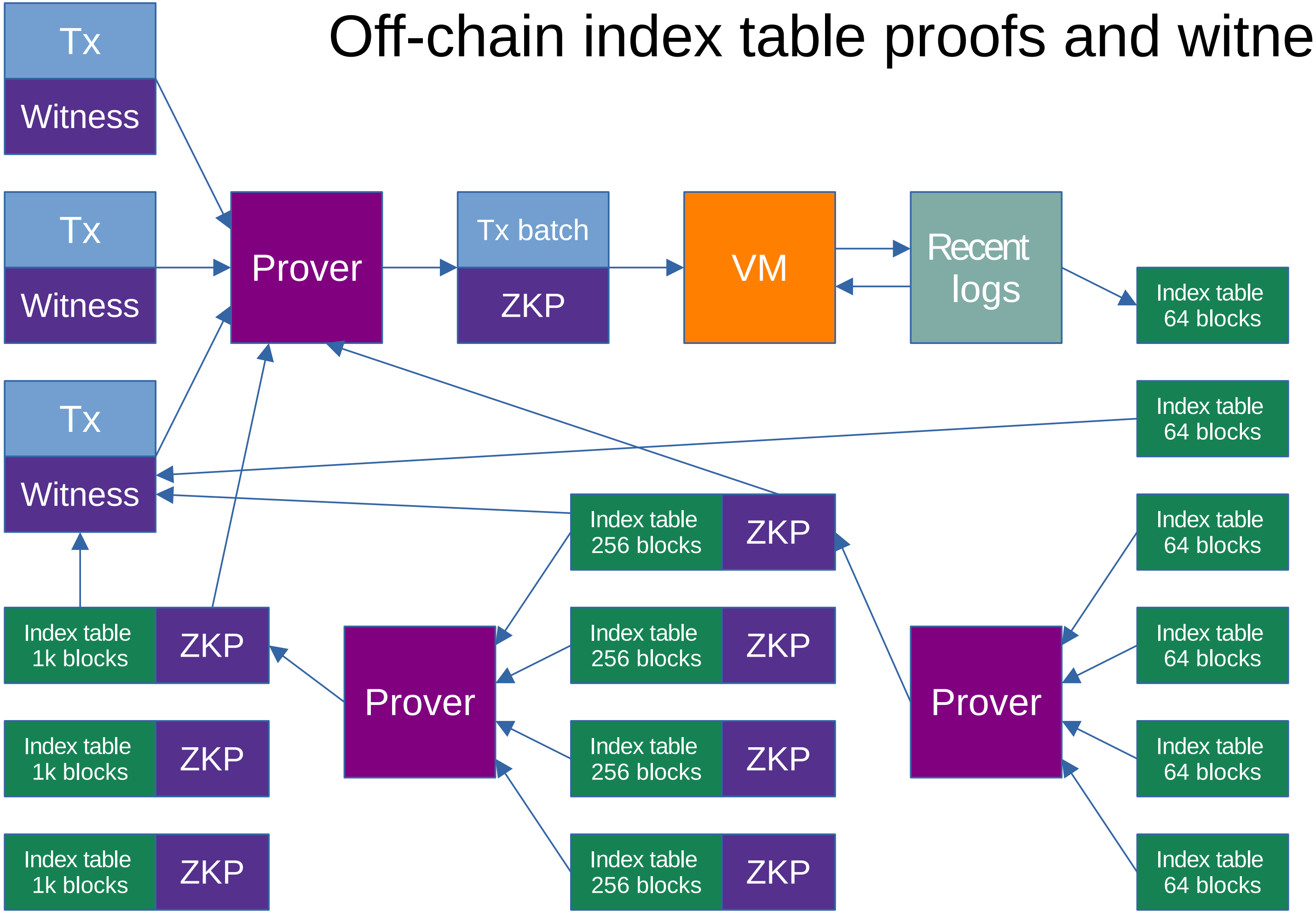
Pro index tables:

- Very cheap to append
- Most of the processing is async/off-chain
- Tables are immutable once rendered
- Easy to distribute, no provider centralization issue (can even add erasure coding)
- Easy to expire/resurrect (no dead tree stubs, old table proofs do not get invalidated)
- Time information is available, contracts can be explicitly designed to search with a limited block range

Multi-level transaction prover infrastructure



Off-chain index table proofs and witnesses



Keys to a healthy infrastructure

- Trustless operation
- Healthy incentives
- Low entry barriers (no bottlenecks, no infrastructural capture risk)
  - No super provers needed
    - Witness check proofs can be processed in small transaction batches
    - Index table merge proofs are recursive, can be created by many provers asynchronously
  - No super witness providers needed
    - Index tables are immutable once finalized, big historic tables can be distributed
  - No super sequencers needed (execution sharding)
    - Needs efficient cross-chain comms (TLI allows log-based CCC)
    - Log-based contract logic is ideal for sharding (already kind of a message based architecture)

Thank you for your attention!

Trustless Log Index repository:

https://github.com/zsfelfoldi/tli

Contact:

zsfelfoldi @ discord

zsfelfoldi@ethereum.org