EIP-7745

Trustless log index

Zsolt Felföldi
zsfelfoldi@ethereum.org

Presented at ACDE #214
19/06/2025

Why?

- Logs are an important part of the current Ethereum ecosystem
- There is practically no way to provide trustless proofs for log queries
- Even local log lookup requires some kind of extra indexing
- Canonical transaction lookup by hash suffers from the same problem
- We should also care about the “UX endgame” – relying on trusted JSON RPC providers kind of beats the whole purpose of Ethereum - we should converge to a future with all-trustless APIs used everywhere by default
- EIP-7745 provides a very efficient lookup structure that can also be tree hashed into a single Merkle root to replace the useless bloom filter field

Finding the right kind of data structure

- With a linear index (like the header chain bloom filters) it is…
  - Cheap to add new items
  - Very expensive to look up a certain item
  - Easy to discard old history
- With a global Merkle tree (like the state tree) it is…
  - Expensive to add new items
  - Cheap to look up a certain item
  - Discarding old entries is complex and inefficient
- Log events and related index data should be…
  - Cheaper to add than state writes
  - Still reasonably efficient to search
  - Easy to discard when “expired”

Filter map properties

- Not a bloom filter any more but still a probabilistic approach that provides high search efficiency
- Fixed size sparse 2D bit maps (fixed density, not attached to block boundaries; automatically scales well with higher gas limits)
- Solves uneven lookup key distribution
- Consistent false positive rates that are easily adjustable with a single parameter
- General purpose hash lookup that gives exact position, allowing efficient pattern matching
- Can also be used to look up transactions by hash (not specified in the EIP yet)
- Cheap to generate, efficient to look up and prove
  - Smart tree hashing scheme: 1024 fixed size filter map trees are combined into a log index epoch
  - Minimal state needed to generate consensus: cca 25Mb
  - Data accessed while searching 10 years mainnet history: cca 1-5Mb
  - First is proportional, second is inversely related to number of map rows

Does it fit the “scaling endgame”?

- ZK proofs might allow much higher gas limits but there is always a bottleneck – and adding log events can still be a lot cheaper at that bottleneck than state writes

- Most of the work needed to update the log index, create and verify proofs is hashing – if the hash function is ZK friendly then the whole technology is

- Data structure efficiency is kind of orthogonal to ZK technology - still easier to create a ZK proof based on 1Mb of data than based on 500Gb

- If ZK tech makes log index proofs even cheaper then it might be a viable option for certain apps to use only logs and no state

Current state of implementation

- Filter map structure fully implemented in Geth and used to serve *eth_getLogs*
  - **https://github.com/ethereum/go-ethereum/tree/master/core/filtermaps**
  - Typical search time for 10 years mainnet history: 300ms to 5s (not optimized yet)
- Consensus hash tree generator exists as PoC on a single-node devnet (the code needed for consensus is <1000 LOC)
- Prover/verifier also exists as PoC
  - Specs and test vectors: **https://github.com/zsfelfoldi/eip-7745**
- Short-term plans:
  - Improve human readable specs for consensus implementation
  - Python execution specs
  - Efficient log index proof format
  - Improve prover/verifier specs
  - API endpoint specification