Vision Statement

Team Name: Avalution
Project Name: Turbocharging Authenticated Storage for Blockchains
Project Sponsor: Ava Labs
Mentor: Aaron Buchwald
Team Lead: Aaron Lee
Team Scribe: Adyah Rastogi

Team Members

- Adyah Rastogi: adyah@ucsb.edu
- Wesley Truong: wesleytruong@ucsb.edu
- Justin Lang: jlang61@ucsb.edu
- Hao(Aaron) Lee: hylee@ucsb.edu
- Jiahua(Roy) Ren: j_ren@ucsb.edu

About the Project

- **What problem is the project solving**
    - Merkle tries utilize databases such as LevelDB, RocksDB in order to store authenticated key-value pairs. However, this requirement adds another layer of database systems which may be inefficient and complex.
    - A golang implementation of Firewood, the Rust implementation of the on-disk merkle trie database.
    - Improving on the efficiency of Firewood through the implementation of serialization, free list, revision manager, and on-disk store
- **Why the problem is important**
    - By creating this merkle trie we make authentication and the database more efficient.
    - This project addresses the problem of efficiently maintaining authenticated state on disk (Tomescu's Challenge).
    - Impactful because 80% of the time spent executing these blocks are in Merkle Tries and grabbing data.
    - There are many blockchain clients who use Golang, and it's easier to call code in the same language/integrate code from the same language.
- **How the problem is solved today**
    - We do not believe that there is a solution to this problem today in Golang.

Outcome of Project

- Utilize the “Firewood” design in order to store key-value pairs in Merkle tries’ data layout for an optimized storage system.
- Produce the first Golang implementation of the Firewood design for efficient merkle trie storage.

Milestones

- Understanding Blockchain
- Learn Golang, Merkle Patricia Trees
- Learn the Firewood project design and AvalancheGo MerkleDB design (golang merkle trie implementation writing to a generic key-value store that can be repurposed for this project)
- Write high level spec for the individual components (serialization, free list, revision manager, and on-disk store)
- Implement on-disk serialization format and disk manager
- Implement revision manager to capture added/deleted nodes and write to disk manager
- Implement free list to re-use disk space from deleted nodes
- Update disk manager to utilize the free list
- Implement on-disk version of the free list
- Perform benchmarking at various database sizes (1GB to 1TB) to evaluate performance
- Optimize performance including memory allocations and write strategy (new-space vs. free list)
- Optional: plug directly into Avalanche C-Chain or Subnet-EVM to compare performance of executing the entire chain on an LSM Tree vs. Firewood
- Implement the state sync functionality, allowing the system to serve queries efficiently from previous revisions.
- Conduct benchmarking to identify and address bottlenecks for iteration

Technologies

- Rust
- Golang
- Firewood
- Avalanche Merkle DB
- LMDB - B+ Tree Based Database
- LMDBX - LMDB used by Erigon and Reth Ethereum Execution Clients