MonadDb

MonadDb is a custom database for storing blockchain state.

Most Ethereum clients use key-value databases that are implemented as either B-Tree (an example is LMDB) or LSM-Tree (examples are LevelD B and RocksDB) data structures. However Ethereum uses the Merkle Patricia Trie (MPT) data structure for storing state. This results in a suboptimal solution where one data structure is embedded into another data structure of a different type. MonadDb implements a Patricia Trie data structure natively, both on-disk and in-memory.

Monad executes multiple transactions in parallel . When one transaction needs to read state from disk, one should not block waiting for that operation to complete - instead one should initiate the read and then start working on another transaction in the meantime. Therefore the problem needs asynchronous i/o (async i/o) for the database. The above mentioned key-value databases lack proper async i/o support (although there are some efforts to improve in this area). MonadDb fully utilizes the latest kernel support for async i/o (on Linux this is io_uring). This avoids needing to spawn a large number of kernel threads to handle pending i/o requests in an attempt to perform work asynchronously.

MonadDb makes a number of other optimizations related to i/o, such as bypassing the filesystem which add expensive overhead.

Previous Parallel Execution Next Transaction lifecycle in Monad Last updated5 months ago

On this page