Hi! I've been working on a project called Quadrable that I think might be of some interest to this group.

It's a sparse, binary merkle tree database. The primary implementation is a header-only C++17 library, and most of the functionality is exposed by a git-esque command-line tool (quadb

). I also have an implementation of the core operations in Solidity. One of the main applications I have in mind is an optimistic roll-up system.

Features:

- Persistent database

: Tightly integrated with the LMDB embedded database, instead of as a stand-alone tree data-structure library. No limits on key or value size. ACID transactions. Instant crash recovery (no write-ahead logs).

- Multi-version

: Many trees can exist in the DB simultaneously, and all structure is shared where possible. Making snapshots or checkpoints is cheap, as is switching between them. Orphaned nodes can be garbage collected.

- Combined proofs

: When making proofs for multiple elements, redundant and computable nodes are omitted (I think this is sometimes also called a multi-proof). Quadrable's approach is a bit more complicated than the usual octopus algorithm since our leaves can be at different heights in the tree. Instead, there is a mini "proof command-language" to hash and merge strands together to re-build the tree. A separation between proofs and the various possible proof encodings is maintained.

- Partial trees

: While verifying a proof, a "partial tree" is constructed (in fact, this is the only way to verify a proof). A partial-tree can be queried in the same way as if you had the full tree locally, although it will throw errors if you try to access non-authenticated values. You can also make modifications on a partial-tree: the new root of the partial-tree will be the same as the root would be if you made the same modifications to the full tree. Once a partial tree has been created, additional proofs that were exported from the same tree can be merged in, expanding the partial-tree over time as new proofs are received. New proofs can also be generated from

a partial-tree, as long as the values to prove are present (or were proved to not be present).

- Appendable logs

: In addition to the sparse map interface, where insertion order doesn't affect the resulting root, there is also support for appendable (aka pushable) logs. These are built on top of the sparse merkle tree, but proofs for consecutive elements in the log are smaller because of the combined proof optimisations. Pushable proofs let you append unlimited number of elements onto partial trees (a pushable proof is essentially just a non-inclusion for the next free index).

Interfaces:

- C++

: Batchable operations. Multiple modifications or retrievals can be made in one traversal of the tree. All get operations are zero-copy: values are returned to your application as pointers into the memory map.

- Solidity

: Supports importing proofs and the core get/put/push operations on the resulting partial trees. All pure functions. Recursion-free. ~700 lines of code.

- quadb

command-line app: 20+ sub-commands. Snapshot checkouts/forking. Batch imports/exports. Tree diff/patch. Debugging and dumping.

Other:

- I'm told the documentation is pretty good. There are some colourful pictures.
- Nearly 100% test coverage. Address sanitiser support. AFL fuzzing of proof decoder started.
- Everything is BSD licensed