We thought this might be interesting to folks here:Reducing state size on Mir.

This is a stateless storage scheme inspired by some old threads here, such asHistory, state, and asynchronous accumulators in the stateless model. To briefly recap the problem: Merkle Mountain Ranges (MMRs) work nicely as a log of account states, but since they are append-only, we need a separate structure to keep track of which states are presently active.

Naively, we could store a vector of “is active” bits. This bit vector might be rather sparse, though, if most states in the log are inactive. Our solution is to apply a run-length encoding, then compress the list of zero-run lengths using Huffman coding.

Since we also want our structure to support fast lookups and authenticated updates, we partially Merklize it. In particular, we store a Merkle tree of “is active” bits, but subtrees beneath a certain height are compressed using the LRE-Huffman encoding. Creating or verifying a Merkle proof involves decoding one of these subtrees and recomputing its Merkle data. There is a time-space tradeoff here, but in practice, we can get pretty close to the compactness of LRE-Huffman and the speed of an ordinary Merkle tree.

The post has some more details, and a proof of concept to show concrete space efficiency.