This proposal is not

introducing any changes to Ethereum; it merely outlines a novel way to write contracts in a way that would result in their on-chain state consisting of a single 256-bit value. Any contract can be algorithmically transformed (i.e. compiled) into this proposed form. Since storing data on the blockchain is expensive and its cost can be expected to be internalized by the introduction of state rent, this proposal can be regarded as a scalability improvement, as it allows Ethereum contracts to manipulate arbitrarily large state without having to store all of it on the blockchain.

Swarm

, for the purpose of this proposal, is just a distributed preimage archive that allows the retrieval of fixed-sized chunks of data based on their crypgoraphic hash, which is (relatively) cheap to compute in EVM.

Contract state

is a 256 bit to 256 bit map.

Contract state can be represented as a Merkleized binary PATRICIA trie in which every inner node (representing a subtrie) contains a bit offset of the branching (i.e. the length of the common prefix of keys of the subtrie in bits) and references to the two subtries. Leaves are simply (K,V) mappings. Note that the number of inner nodes is always exactly one less than the number of mappings. Each insertion except the first one results in adding exactly one inner node and one leaf; conversely, each deletion except the last one removes one of each. Note that there are ways of representing binary PATRICIA tries in such a way that each node contains one mapping, one bit offset and only one reference to another node; the actual implementation of the proposal might opt for using such a representation.

If the reference to a node is its hash value, then an insertion or a deletion can alter at most 255 nodes, irrespective of the size of the map. In practice, keys in the state are typically hashes of the actual values used as keys, which makes the trie representation balanced and its depth the logarithm of the map's size. In this typical case, insertions and deletions alter a logarithmic number of nodes.

Thus, one can replace SSTORE and SLOAD opcodes by subroutines that perform the corresponding operations on an externally stored Merkleized map, using the root reference stored in the contract's onchain state and additional data with Merkle proofs supplied in transaction data. The replacement of SLOAD will return the value supplied in transaction data, but will also verify the Merkle proof that the K,V mapping (including implicit K,0 mappings) is, indeed, present in the map with the on-chain root reference. The replacement of SSTORE will update the on-chain root reference.

Note, furthermore, that the pre-image oracle necessary for the use of such contracts can be instantiated using only transaction history, though in practice it is, of course, much more convenient if an efficient Swarm-like content-addressed storage network performs this role.

It is also important ot note that in this model, at most one of concurrently submitted transactions altering the state can succeed; the ones that fail need to be re-submitted with Merkle proofs calculated from the state updated by the successful transaction.

Essentially, this proposal virtualizes the stateless client model on top of current EVM, thus reducing the on-chain contract state to a single Swarm-reference.