TLDR

: We outline considerations for a native integration of Eth1 into an Eth2 shard.

Single shard

Below are possible work items to “premine” the Eth1 consensus and state into a shard.

1. BLS vs ECDSA

: Add BLS12-381 opcode or precompile to EVM to verify signatures of Eth2 consensus objects.

1. Keccak256 vs SHA256

: Reduce gas cost of calls to SHA256 precompile (and possibly to all precompile calls) to facilitate checking Eth2 consensus Merkle paths.

1. roots

: Add EVM opcode to read beacon chain block roots (alternatively, an opcode to read state roots). This could be useful, for example, to prove that a given Eth1 block has been finalized without adding an ad-hoc finality opcode.

1. SSZ vs RLP

: Add support for SSZ and consider deprecating RLP.

1. sparse tree

: Consider removing the hexary Patricia tree in favour of a sparse binary tree.

1. Casper FFG

: As part of an intermediate hybrid PoW/PoS step, update PoW fork choice rule to respect beacon chain finality.

1. light clients

: Replace PoW-based Eth1 light clients with PoS-based beacon chain light clients.

1. randomness

: Find a migration path for dApps that rely on the nonce

header field for randomness. RANDAO only provides a decent random number every epoch, not

every block as with PoW.

1. Eth1 headers

: Consider redefining Eth1 header fields which may no longer be relevant (e.g. ommersHash

, difficulty

, beneficiary

, nonce

, gasLimit

) and make sure no significant dApps break as a result.

1. incentivisation

: Remove PoW rewards.

1. difficulty bomb

: Remove difficulty bomb.

1. statelessness

: Build statelessness support into Eth1 clients (Geth, Parity, etc.). Also implement logic to produce Eth2-compliant shard blocks.

1. new liveness invariant

: Make sure that there are no significant dApps that break when moving to regular 3-second slots (as opposed to the current Poisson distribution for blocks, with a ~15sec average block time).

1. libp2p vs devp2p

: Migrate Eth1 clients from devp2p to libp2p.

1. Eth1 gas limit

: Consider removing the Eth1 gas limit to avoid having two gas limits (Eth1 gas limit under Eth2 gas limit).

1. Eth2 gas limit

: Make sure that 8,000,000 “Eth1 EVM gas” when translated to “Eth2 WASM gas” fits within the Eth2 gas limit. Significantly reducing the effective limit below 8,000,000 gas may cause unacceptable congestion. Check that the Eth2 gas mechanism (which would not have miner voting) is otherwise acceptable for Eth1.

1. WASMify Eth1 consensus

: Formalise the Eth1 state transition function—including all precompiles—as WASM code. Split the WASM code into small chunks that fit within the Eth2 code limit (possibly 16kB per code chunk). This formalisation risks introducing consensus bugs on a $20B+ network so extensive fuzzing and formal verification may be required. We also want the WASM formalisation to be consistent with the wider WASM-for-blockchains standardisation effort.

1. premine load imbalance

: Consider the negative consequences of the load asymmetry from premining the Eth1 state into a single shard with the other 1023 shards empty.

1. hard forks

: Negotiate a timely hard fork schedule with the existing Eth1 ecosystem.

1. Eth2 consensus pollution

: Consider disruptions to Eth2 whenever Eth1 will need to hark fork. Indeed, the Eth1 consensus would likely have “too big to fail” status within the Eth2 ecosystem (inconsistent with the expectation that execution engines are application-layer, not consensus-layer) which imposes an Eth2 hard fork for every Eth1 hard fork.

All shards

Below are further considerations to put Eth1 consensus logic on all shards.

1. state growth

: Make sure that the ecosystem can handle 1024 privileged copies of the Eth1 consensus. In particular, the current no-rent approach may make the maintenance costs of tracking all shards too high for services such as exchanges and Etherscan.

1. shard number

: Consider adding a shard number field in Eth1 headers to easily differentiate Eth1 execution engines running on different shards.

1. yanking

: Add support for contracts to move between shards and make sure that the asynchronisity introduced does not break existing Eth1 dApps.

Conclusion

Safely integrating Eth1 into Eth2 is a significant engineering and governance effort. It seems doable on a long-enough (likely multi-year) timescale. The native integration should be compared to significantly cheaper medium-term alternatives. For example, a two-way bridge between Eth1 and Eth2 can be built using light clients.