Summary of scalability challenges

The current limitations on the transactions-per-second (TPS) of Ethereum are complicated and have no simple solutions in the general case, and unlike common misconceptions of the bottleneck being on consensus side or p2p or the choice of VM, the long term bottlenecks are a trilemma of:

- the cost of accessing ever-growing storage,
- unpredictability of arbitrary turing-complete execution
- and bandwidth cost + latency of block and tx propagation

Of the three, the one with the most major impact on the current Ethereum mainnet is storage cost - the scalability ceiling of which is limited by the physical constraints of current hardware, like the max IOPS of NVMe SSDs, buffer sizes on the disk's memory, and bandwidth/latency sensitivity between the disk and RAM/CPU pipeline. Increasing the (stateless) execution speed is only a small part of the this equation, as no matter the backend used to store data, there's no way to have constant time key/value lookup for an ever-growing dynamic dataset - meaning that the cost of state-lookup dependent opcodes increases monotonically over time. Given that state access cannot be parallelized and block production is highly latency sensitive, unrestricted computation over this state poses a significant scaling challenge.

The following graph created by Martin Swende of the EF shows what dominates most of the EVM execution time in a geth full sync, and its increase in impact as the trie sizes get bigger:

[

1805×1284 215 KB

](https://forum.celestia.org/uploads/default/original/1X/65df68b4200ae2d22f8c52ae1509b01a2bb11adc.png)

Observing the legend at the bottom, it is easy to see that what increases the most over time is the cost of storage_reads

and account_reads

. Therefore, the overall block gas limit in Ethereum right now is most heavily influenced by worst case scenario "long-term attack" of someone bloating the state as much as possible by using 30m gas worth of state updates per block.

There are research endeavours and proposals to deal with the storage overhead and other limits in a more comprehensive way, such as separating the cost of storage from the cost of execution, and bandwidth, rather than pricing all of them in the same way through the same gas[1]

.

Optimizations for application specific EVM chains

However, for application specific instantiations of EVM chains, such as a rollup settlement hub, there are shortcuts that can be taken to heavily amortize this cost and remove many of the worst case scenario attacks that require conservative limits on the execution to mitigate.

Restrict contract set to prevent state bloat

In the case of using the EVM as a settlement hub, and restricting the type of state accesses/writes by having a fixed set of well analyzed bridge contracts, enables limiting the amount of writes that can happen in a tx independent of its gas limit, and as a result greatly increase TPS without worrying about nodes not being able to keep up or state bloating to the point of nodes becoming next to impossible to reliably run (as has happened with BSC[2]

).

Reduction of state access cost

The cost of state access on the current version of Ethereum protocol is due to the large size of the current state trie, which is over 150GB, holding tens of millions of key/value pairs. By starting a new chain which has contract-enforced restrictions on how much state can be accessed in a single block or transaction, and limits how much state can grow per block, we can reduce the cost of state access by at least 50% and keep it low for longer than mainnet has been able to.

Cache improvement through predictable access

One storage level improvement that is easily leveraged and already implemented in the EVM is "access lists"[3]

- a cache/disk access optimization where you specify what addresses and storage keys the tx will hit, and those accesses become cheaper - but in case there is a cache miss then they revert to being more expensive. In normal turing-complete-contracts-can-call-anything EVM it's very difficult to predict these storage keys ahead of time in the general case, but within the restricted hub it becomes possible to make state access deterministic and safely increase tx throughput.

Contract preloaded in full node and light client software

A type of state access that has higher load impact than a normal SLOAD

from storage is the act of calling into another contract. At time of writing, the latest Ethereum protocol version allows a contract's code to be up to 24kb in size - meaning that any *CALL opcodes or any that force the reading of that state, like EXTCODEHASH or EXTCODECOPY, can cause the read of up to 24kb of data - which increases the load impact for light clients as well, as they won't have to just request the merkle proofs to N

storage slots, but potentially also K

24kb contract bytecodes. This can become large overhead on the network, given that the external account call opcodes are as cheap as 2600 gas - far less than the cost of loading 24kb with SLOAD

, which would be 1.6 million gas.

By restricting users from being able to deploy arbitrary contract, this attack vector is also removed, as the fixed set of deployed and deployable contracts can be preloaded in the genesis file, or in an auxiliary store that is updated only with updates to the protocol version. Hence, a light client will always have the code for a call destination by default, and won't ever have to request it from the network. Even in the case where we allow users to create new contracts through a Factory contract, it's possible avoid the propagation overhead by requiring the codehash stay the same (restricting the factory to only initialize state). The tx can include a codehash in the access lists that corresponds to which preloaded bytecode was deployed at that address.

Delegated Security and Data Availability

There is of course the third pillar of the trilemma, being the propagation latency impact of transmitting and processing large blocks and transactions, but that is an ephemeral cost - propagating a large block has the potential to slow down the network only at the moment of propagation, but will not necessarily be a burden to all future nodes. For example, if the block is simply a rollup committing large amounts of calldata by storing the hash, the calldata cannot be reference by a smart contract and so one can't force full nodes to retrieve it via tx execution. So it is a block with latency impact at the moment of propagation, but low execution, storage, and sync cost. These ephemeral costs are more easily dealt with, especially in chains that do not need to have as many nodes for security, like ones built on Celestia that delegate security and data availability to the Celestia chain.

Additionally, having a restricted domain in the space of possible transactions prevents the decision problem that would prefer rollup txs above all else and price out regular txs, as stated by geth team's comments on EIP-4488[4]

:

Specifically, we fear that EIP-4488 will, due to the 2-dimensional nature of the scheme,

favor rollup transactions so much that it won't be possible for non-rollups to use the

blockchain. The EIP should present significantly more evidence that this will not be the

case.

At this time, EIP-4488 attempts to keep usage balanced by allowing transactions smaller

than 300 bytes even when the block size limit is already reached. While the exception will

allow for simple value transfers to be included alongside a data-heavy rollup transaction,

any transaction larger than 300 bytes needs to outbid all rollups for block space.

Since the settlement hub is specific only to rollup txs and supporting committments, this is does not present a mechanism problem for the hub design.

An application specific EVM can be fraud proofable

All the optimizations above combined further enable another desirable feature: compact fraud proofs via intermediate-state-roots (ISRs) that can be fed to light clients, without light clients needing to potentially request 100MB+ of state to be able to check the fraud proof against two state roots. The restriction on writes and access, contract deployments, and predictability of how much state a tx can access regardless of its gas limit enable these efficient single-round intermediate state root fraud proofs, and remove the 2/3ds honest majority requirement for light nodes - which is important as an application-specific EVM chain/hub would likely have a lower total number of validators or block producers than a chain that needs to provide its own consensus security like Ethereum.

Scalability + Flexibility + Easy Integration & Extensibility

By using an EVM chain that is application-restricted/application-optimized, we gain the best of both worlds, flexibility of a VM chain whose behavior can be upgraded just through writing smart contracts and enabling them with a protocol upgrade (e.g. by DAO vote), and having those contracts be compatible with the broader EVM ecosystem of standards, tooling, libraries, etc, but with the capability to neutralize the biggest problems that come with completely unpredictable execution.

Due to the above optimizations, an application-specific EVM tailored to act as a hub would be able to scale far better than the unrestricted Ethereum protocol, likely to 2 or 3 orders of magnitude higher TPS for rollup txs per the same amount of block gas.

References

[1] Should storage be priced separately from execution? - Economics - Ethereum Research

[2] BSC is a lost cause · Issue #553 · bnb-chain/bsc · GitHub

[3] EIP-2930: Optional access lists

[4] HackMD - Collaborative Markdown Knowledge Base