Scaling Autonomous Worlds

Building the foundations... and sewers
for millions of inhabitants.

tdot@lattice.xyz

What’s on the menu

Context

Who are we and what problem are we trying to solve?

Design patterns

Explore problems faced by MUD apps and the design space to solve them.

Pushing the boundaries of latency

How these design patterns led us to a new system we’re currently prototyping.

Lattice

A full stack Ethereum company

Autonomous Worlds

Extending the vision for what fully onchain games can enable.

MUD

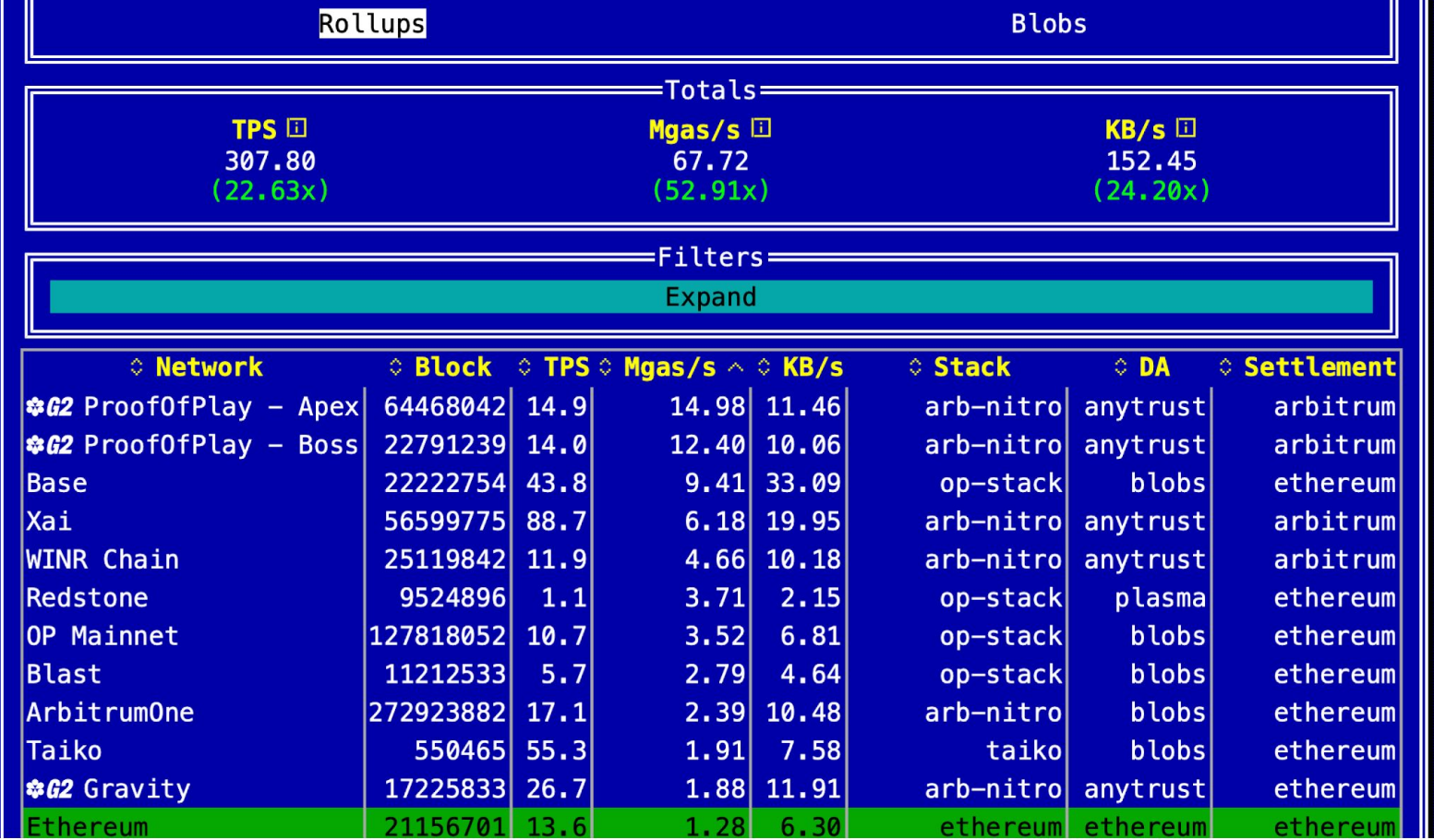
Solidity framework to simplify the creation of complex onchain worlds.

Redstone

Ethereum L2 adding cheap Alt-DA to the OP Stack.

Platform

Running the chain, RPC and indexer services so builders can focus on their worlds.



Onchain games are gas guzzlers

Applications

EVE Frontier

Large team. Lot of custom systems. Physics are offchain. Progressively bringing more logic onchain.

Biomes

Small team. Would rather focus on building the world than running infra. World physics onchain from day 1.

Sewage hackers

Requirements

Real time multiplayer

Writes can propagate fast enough so multiple clients have the same view of the state.

Low gas fees

Users can interact with the world without paying expensive transaction fees.

Ethereum/EVM compatible

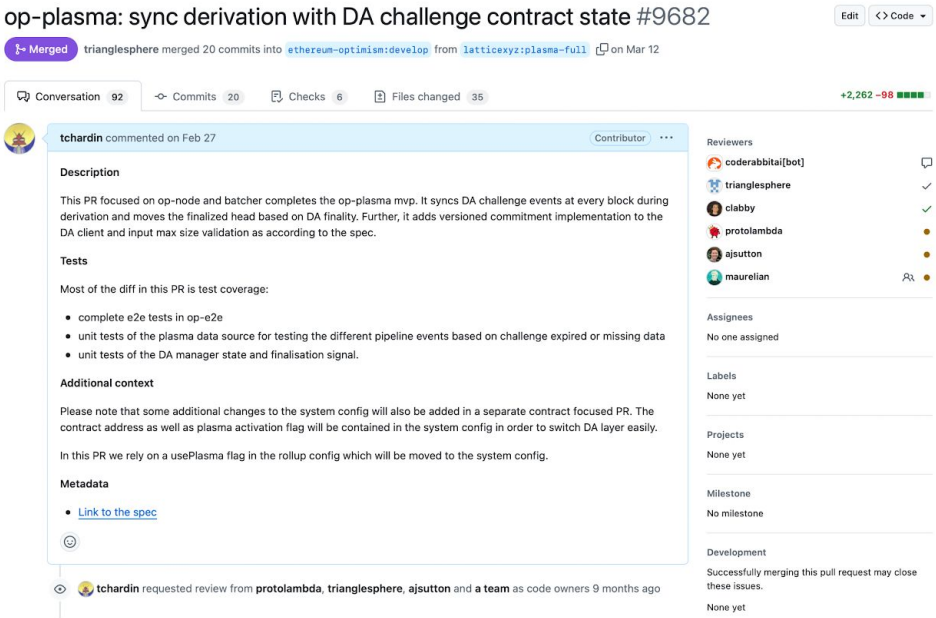
No changes to the Ethereum protocol.

MUD native

Easy to use and integrate for MUD builders.

Redstone

Scaling data availability with op-plasma

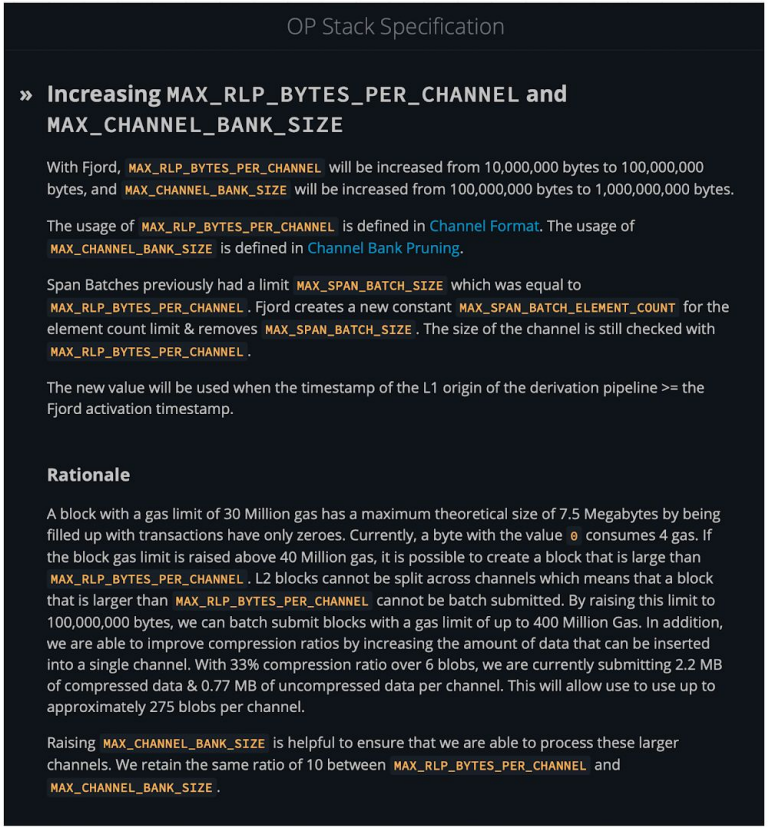


Keep the L1 data fee cheap even as blob space fills up.

Redstone

Increasing OP-Stack derivation throughput

MUD application higher calldata usage uncovered the need for larger rollup derivation payload size limits during load testing.



OP Stack Specification

» Increasing `MAX_RLP_BYTES_PER_CHANNEL` and `MAX_CHANNEL_BANK_SIZE`

With Fjord, `MAX_RLP_BYTES_PER_CHANNEL` will be increased from 10,000,000 bytes to 100,000,000 bytes, and `MAX_CHANNEL_BANK_SIZE` will be increased from 100,000,000 bytes to 1,000,000,000 bytes.

The usage of `MAX_RLP_BYTES_PER_CHANNEL` is defined in Channel Format. The usage of `MAX_CHANNEL_BANK_SIZE` is defined in Channel Bank Pruning.

Span Batches previously had a limit `MAX_SPAN_BATCH_SIZE` which was equal to `MAX_RLP_BYTES_PER_CHANNEL`. Fjord creates a new constant `MAX_SPAN_BATCH_ELEMENT_COUNT` for the element count limit & removes `MAX_SPAN_BATCH_SIZE`. The size of the channel is still checked with `MAX_RLP_BYTES_PER_CHANNEL`.

The new value will be used when the timestamp of the L1 origin of the derivation pipeline >= the Fjord activation timestamp.

Rationale

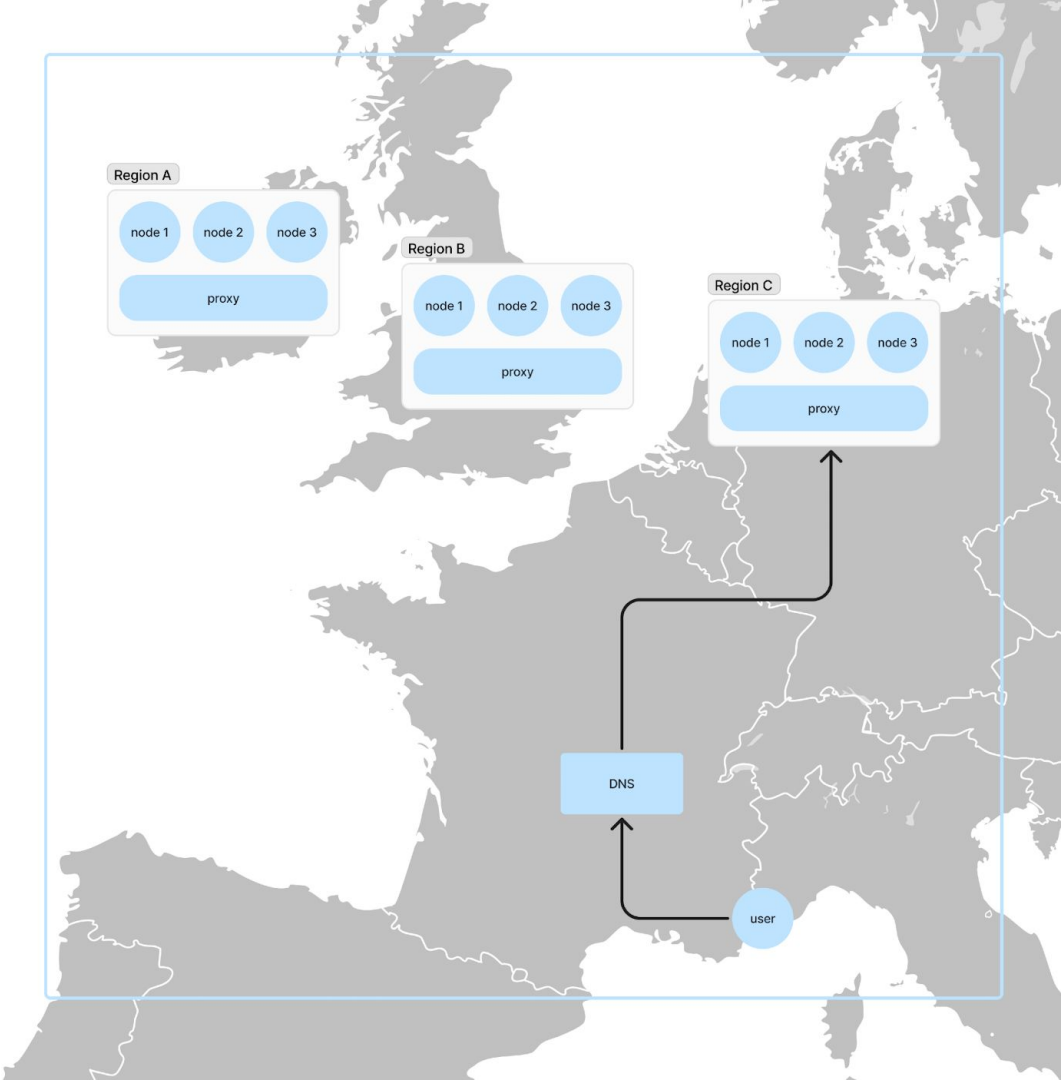
A block with a gas limit of 30 Million gas has a maximum theoretical size of 7.5 Megabytes by being filled up with transactions have only zeroes. Currently, a byte with the value `0` consumes 4 gas. If the block gas limit is raised above 40 Million gas, it is possible to create a block that is large than `MAX_RLP_BYTES_PER_CHANNEL`. L2 blocks cannot be split across channels which means that a block that is larger than `MAX_RLP_BYTES_PER_CHANNEL` cannot be batch submitted. By raising this limit to 100,000,000 bytes, we can batch submit blocks with a gas limit of up to 400 Million Gas. In addition, we are able to improve compression ratios by increasing the amount of data that can be inserted into a single channel. With 33% compression ratio over 6 blobs, we are currently submitting 2.2 MB of compressed data & 0.77 MB of uncompressed data per channel. This will allow use to use up to approximately 275 blobs per channel.

Raising `MAX_CHANNEL_BANK_SIZE` is helpful to ensure that we are able to process these larger channels. We retain the same ratio of 10 between `MAX_RLP_BYTES_PER_CHANNEL` and `MAX_CHANNEL_BANK_SIZE`.

RPC Proxy

Scaling the reads

We can add multiple nodes in different regions and balance RPC request load between them.



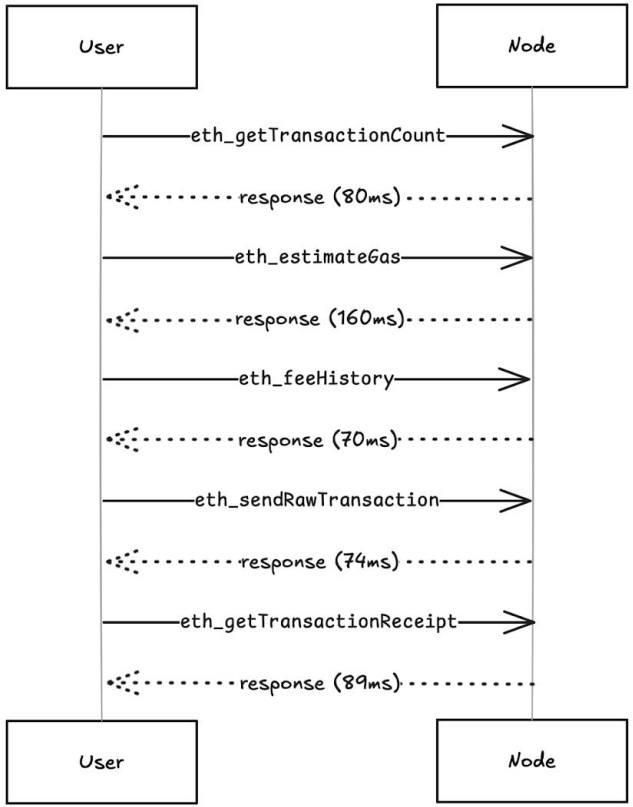
Indexer

Load the initial state faster from an indexer database

Only query what you need in order to display the app's initial load from a postgres database replicated in multiple regions. In collaboration with indexsupply.com.

MUD Client

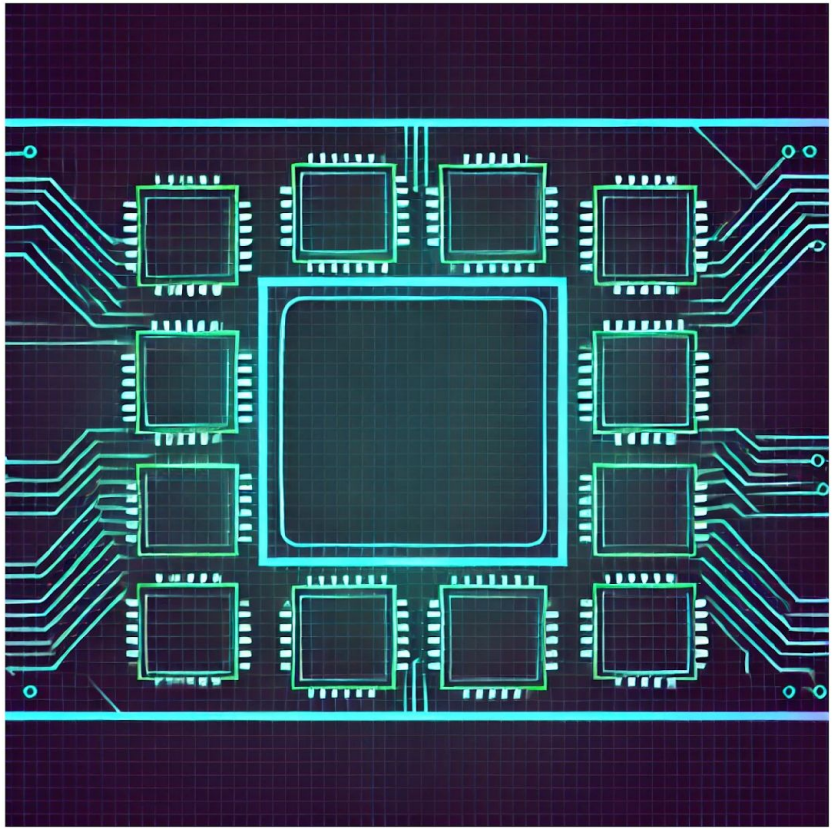
Optimising the browser client



Reduce the number of roundtrips by pre-loading and caching the gas and tip values.

Block Building

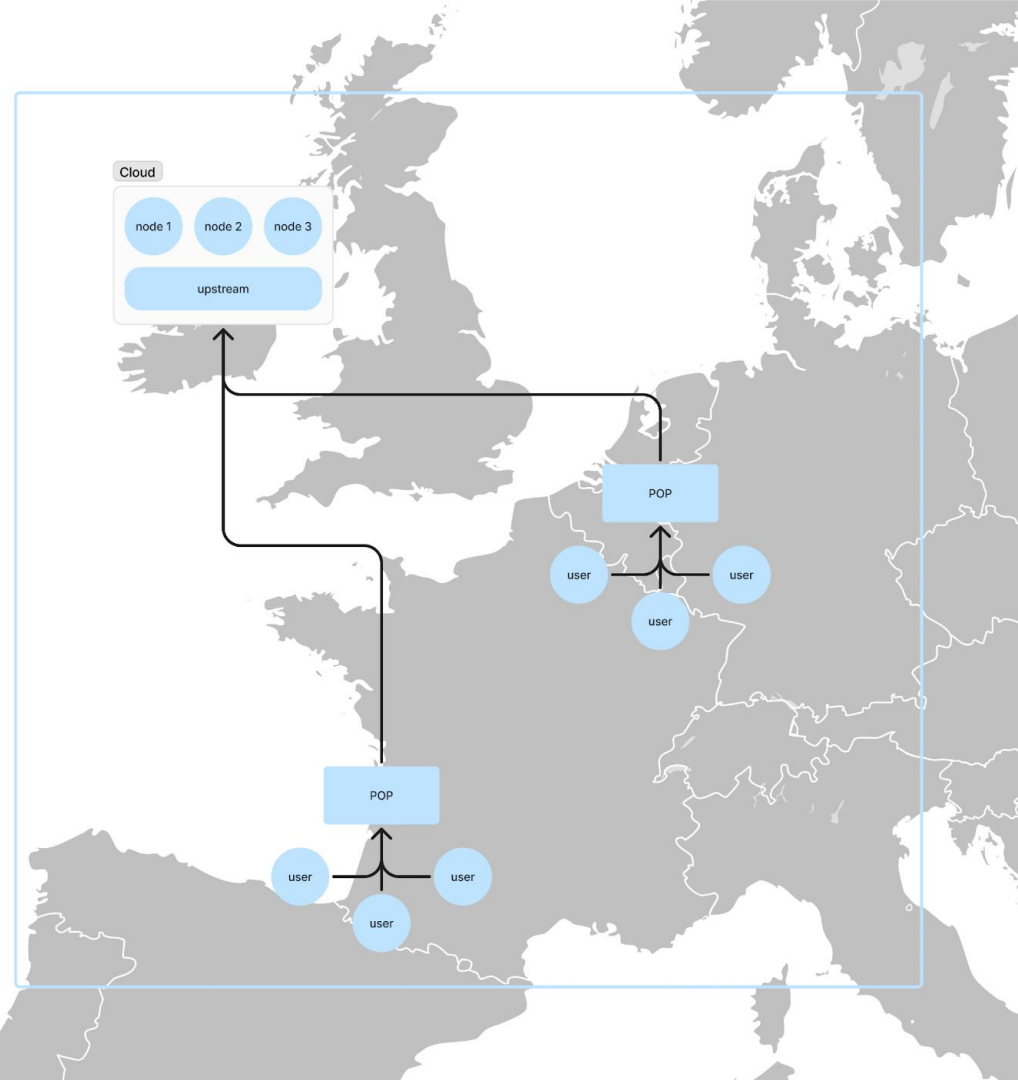
Scaling the writes



Can we speed up block building?

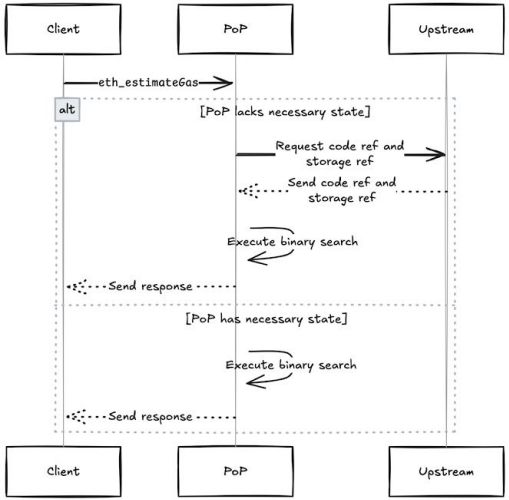
Edge

The MUD CDN



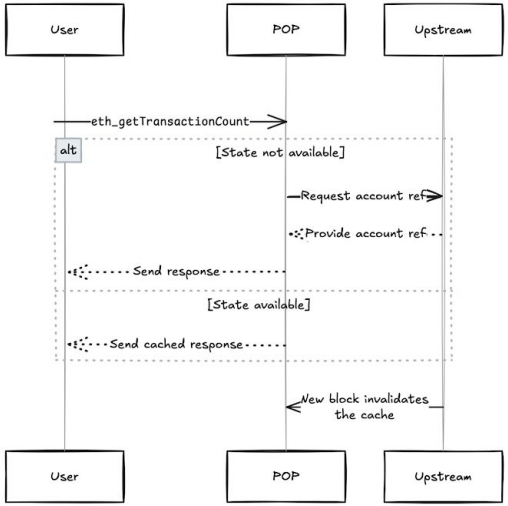
Execute transactions at the edge.

Things we can speed up



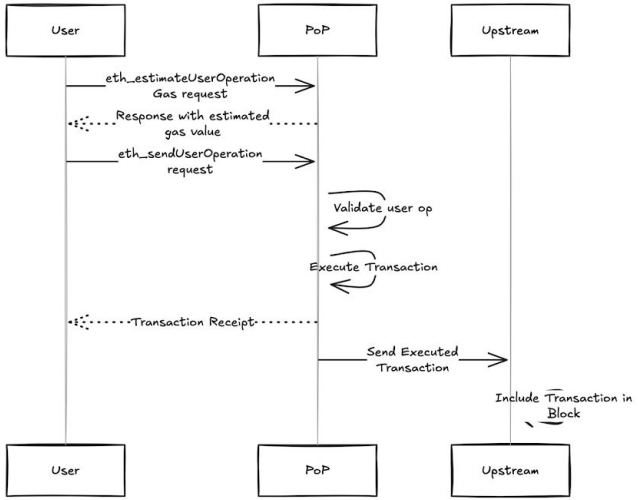
Estimate Gas

The point of presence only pulls the state it needs when the EVM executes the transaction during binary search.



Nonce management

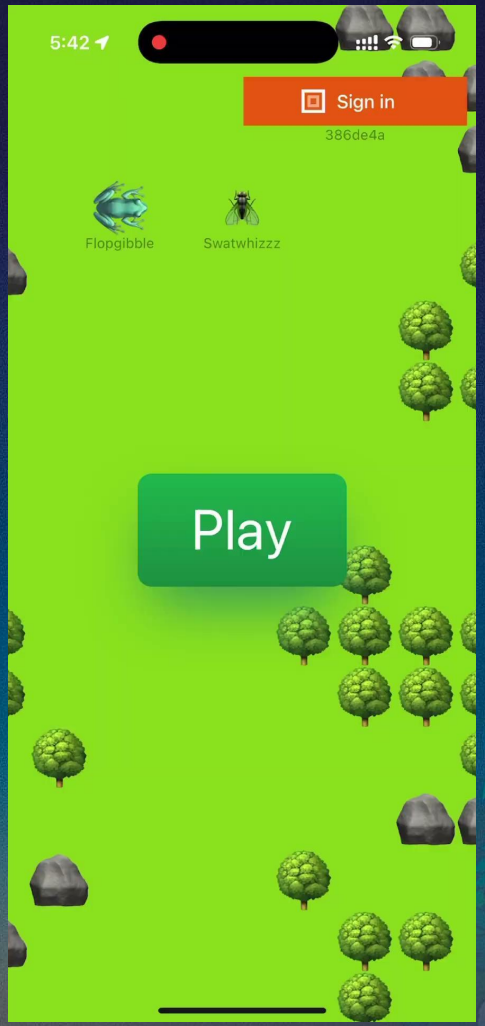
Caching and updating the account state on the edge increases speed when fetching the next nonce.



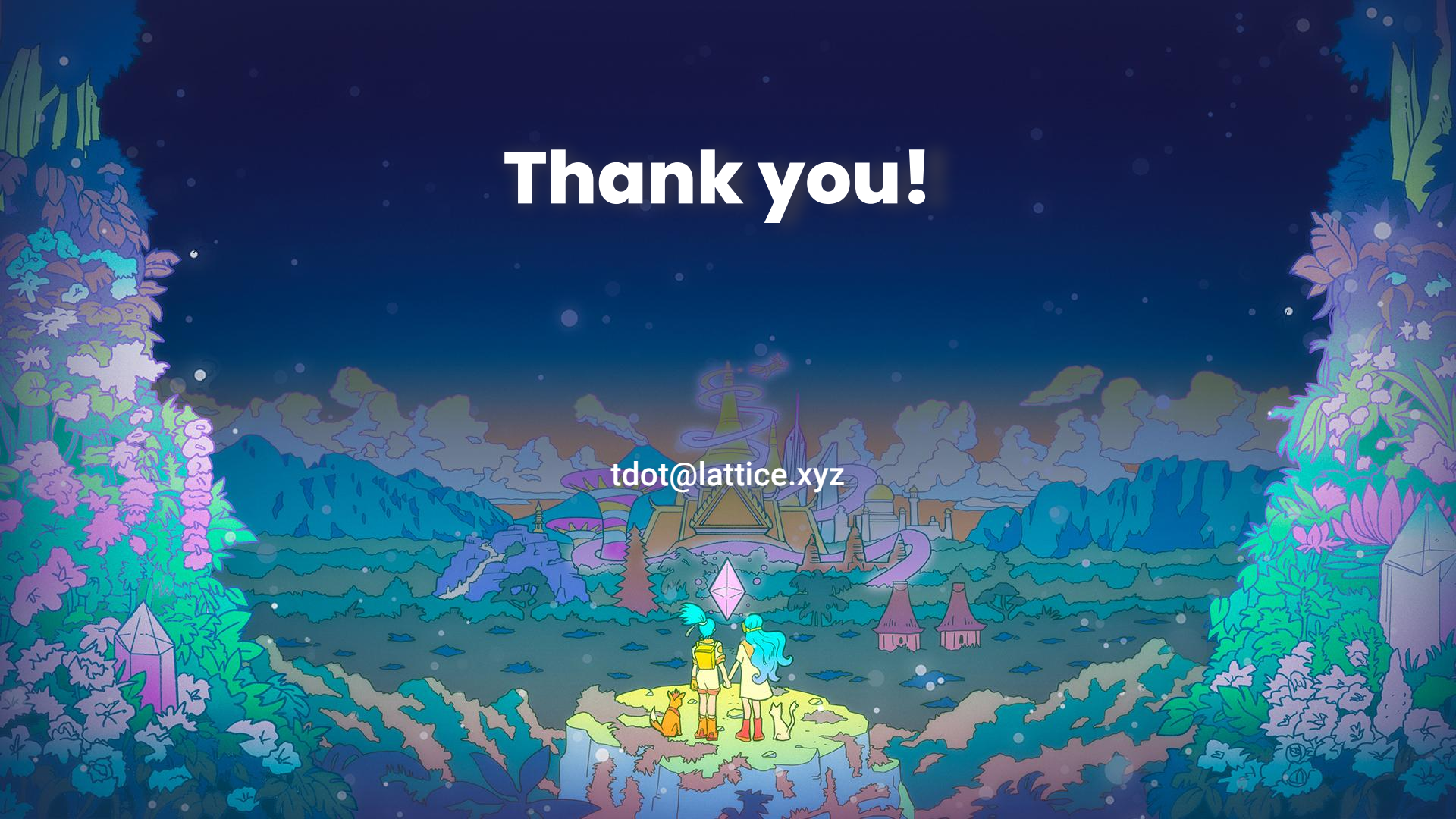
ERC4337 user op bundling

Validate user operations and sign transactions directly on the edge.

Demo



Thank you!



tdot@lattice.xyz