Hi everyone, I would like to introduce a project we worked on during ETH Global Istanbul to you. We call it the Blob Merger, a solution designed to optimize blob data usage on Ethereum after EIP-4844 will be implemented. Since blobs maintain a fixed size, not every user utilizes the entire capacity. Our solution processes submitted blob data, creating efficient blobs that maximize available space.

What is the Blob Merger about

We are building on the Suave blockchain, and our SUAPP consists of two components. First, a precompile responsible for merging the supplied data. Second, a regular app contract that manages the connection between users (typically rollups) and block builders. Users can submit their blob data to the app contract on the Suave blockchain, and builders can use the contract to receive optimized bundles.

You can find more information on the ETH Global page:Blob Merger Showcase or GitHub page.

Future plans

However, during the 36 hours of the hackathon, we only created a concept of the Blob Merger that will be completed in the near future. To take blob merger from a hackathon project into an actual app that is usable by the rollups, the next steps need to be taken:

- Polished SUAPP

: During the hackathon, we've managed to create and write tests only for the precompile. There remains work on the SUAPP contract, tests for the contract, and e2e tests for the app.

- Test on the testnet

: The Blob Merger is tested on the local blockchain. Once the SUAPP is polished (see step one), we need to test it on the testnet as well.

- Fees

: Sending the request to merge blobs requires payment of a transaction fee. To ensure fairness, we propose that the fees be divided among entities that have data in the merged blob, in a ratio corresponding to the data volume. In the event of higher traffic on the network, there will be a greater demand for space in the blob. For such cases, we are considering implementing an auction system so that the rollup can choose whether to wait or pay a premium to expedite the processing of their data. The payment mechanism itself could be resolved through a deposit, but this is still to be discussed for the production version.

- Splitting the data

: Currently, the merger takes blobs and tries to fit them into one merged blob without splitting it into more blobs. So, when we have a merged blob of 117 kB, there is still 11 kB remaining. However, if there isn't any blob sent by the rollup with the calldata size of 11 or smaller, it remains empty. For greater effectiveness, we suggest splitting the data between merged blobs. For example, if there are merged blobs of sizes 117 kB and 123 kB, and in the stack is still the rollup blob of size 13 kB, 11 kB will be added to the first blob, making it 128 kB in size, and the remaining 2 kB to the second blob, making it 125 kB in size.

- Communication back to the user

: The future blob merger should provide a way for the rollups to know if their data was included in a blob.

- Data expiration

: In the future version we would like to add the possibility of expiration of the submitted data and also the ability for the rollups to cancel their request for specific data to be included in a blob.

- Explorer

: We realize there will be a need for a blob explorer as well. We plan to check existing solutions or create our own.

- Standardization

: Write down a standard for how the blob merging should function.

If you have any thoughts or ideas on how to improve the current state or the next steps, we will be happy to discuss it with you. Finally, we would like to thank all the people who helped us during the hackathon, especially Dmarz and the Flashbots team!