Intro

Reddit recently deployed a token airdrops + transfer on ethereum test net. The airdrops costs 99% of the gas. So it makes sense to optimize this specific use case.

The most promising scaling approaches at the moment are ZK and Optimistic rollup. The major on limitation for both of these is the amount of data they need to put on chain. In both cases they need to place the diff between the current state and the next state so that anyone is able to reconstruct the whole state of the system. So representing these diffs in as compressed way is desirable.

We have data for reddit for 3 months. We use this data to test compression.

The code that accompanies this can be found https://github.com/barryWhiteHat/reddit_airdrop_compression

Naive Cost Estimates

Okay first we need to set some naive costs. So assuming a ultra efficient rollup the gas cost is just the data availability cost.

The data cost on chain for airdrop are to (17 bits), amount (16 bits)

so every airdrop costs 33 bits of on chain data.

gas_cost_per_bit = 2 gas_cost_misc ~= 5 gas_cost_total ~= 7

gas_cost_total * 33 = 231 gas per airdrop.

In the data (bricks) from May, June and July we have to do 79724 airdrops. So the total naive cost of airdrop is 79724 * 231 ~ 18M gas.

Example

We had an airdrop like {Alice:1, Bob:1, Carol:1}

In the naive approach we did

{Alice:b0000000000000001 , Bob:b0000000000000001, Carol:b0000000000000001}

In the improve approach we say

{{Alice, Bob, Carol} :b0000000000000001}

which saves us roughly 20%. The maximum we can save is 50% when every user gets the same amount.

The other 50% can be removed if reuse the same groups.

Compress amounts

First lets arrange users into groups who received the same amounts. Then we can compress the amount from each airdrop of the same value which is 16 bits and infer it from some commitment in the batch.

This results in

14264019 total gas naive is 18416244 which is ~22% saved

Reusing groups

So we already have a group in May for Alice Bob and Carol. If in June they each get 1 token again we can redo it for free. Basically just reference the previous round and apply it again. So lets try this in practice.

Repeat users 18,546 which accounts for 34.36% of total users.

In may 25,393 user revised funds. In June 5105 of these same users received funds again. In May is cost us 25393 * 17* 2 = 1M gas to make the data available.

If we made the data available for the 5105 user in June it would cost us 5105 * 17 bits * 2 gas per bits = 173570 gas.

To take the group of 25393 users from may and permute it costs us 25393 * 1 bit * 2 gas per bit = 50786 this is a saving of 30%.

We applied this to the total data available for May, June and July and reduced 3.26 % reduction over naive.

The reason for this relatively modest reduction is that the 30% repeat users is very small compared to the ~95% amount re-usage.

The trend of repeat users is increasing by ~20% per month. So assuming this trend continues we should see a deeper drop in the amount of data put on chain.

Future work

1. Make groups of people who received a similar amount
2. Explore how to make the groups so that it is optimal for future rounds.
3. Add more months of data to the analysis
4. Get future data and try and figure out if it is optimal.
5. Try other approaches to state compression

Conclusion

Given the data we have for 3 months of reddit competition we reduced the on chain data by 26.6 %.

The maximum reduction 100% if we want to send the same airdrop to the same group multiple times then we don’t need to put any data on chain for the second round. We just need to reference the first round.

As more data is released I believe we will improve on 26.6%