

Sprint 2

Check for token_id Details

t table: join transactions and token_transfers table

```
with t as (
select tk_tr.value token_id, tr.block_timestamp,
    tk_tr.block_timestamp, receipt_gas_used,
    (receipt_gas_used*0.0000036756792144+66.1136)*20/3600 *0.0004 watt_per_tran
FROM `bigquery-public-data.crypto_ethereum.transactions` tr
  join `bigquery-public-data.crypto_ethereum.token_transfers` tk_tr
    on tr.`hash` = tk_tr.transaction_hash
where tr.to_address = lower('0x1a92f7381b9f03921564a437210bb9396471050c'))
```

Strange token_id

```
select distinct (token_id) from t
where length(token_id) >= 5
```

token_id
5000000000000000000
3000000000000000000
95000000000000000000
2000000000000000000

Total number of transactions with strange token_id

```
select count(*) total_tran_with_strange_token from t
where length(token_id) >= 8
```

total_tran_with_strange_token
126

token_id ≤ 5 with hash

```
select token_id, transaction_hash, block_timestamp, tr_from, tr_to,
    tk_tr_from, tk_tr_to, receipt_gas_used, watt_per_tran,
    dense_rank() over(partition by token_id order by block_timestamp) rn
from t
where length(token_id) <= 5
order by token_id, transaction_hash
```

token_id	transaction_hash	block_timestamp
0	0x9b4c1a1e0e72d41b85c2fec185433f95b63fa27178c86086f961844f9b59a90a	2021-07-03 17:34:34 UTC
10	0x00c1f708e2826aad71beac79c8b777d96aed6a0aef842d69d46bd00050ccf422	2021-07-01 10:31:24 UTC
10	0xab3a528d28098cbd3267d54dc5e94687146bbd1935babc920c739e3d4fdf7df6	2021-07-29 10:34:06 UTC
100	0xaf226d56128eb304b44658d459e265595403087612e3ae8f359044158ca27800	2021-07-01 16:59:19 UTC
1000	0xdc27b766142da84d033bc613331caa233640f8ba9007e2c0129e0ca7b80e4216	2021-07-01 18:29:04 UTC
1001	0xdc27b766142da84d033bc613331caa233640f8ba9007e2c0129e0ca7b80e4216	2021-07-01 18:29:04 UTC
1002	0xdc27b766142da84d033bc613331caa233640f8ba9007e2c0129e0ca7b80e4216	2021-07-01 18:29:04 UTC
1003	0xdc27b766142da84d033bc613331caa233640f8ba9007e2c0129e0ca7b80e4216	2021-07-01 18:29:04 UTC
1004	0xdc27b766142da84d033bc613331caa233640f8ba9007e2c0129e0ca7b80e4216	2021-07-01 18:29:04 UTC
1005	0xdc27b766142da84d033bc613331caa233640f8ba9007e2c0129e0ca7b80e4216	2021-07-01 18:29:04 UTC
1006	0xdc27b766142da84d033bc613331caa233640f8ba9007e2c0129e0ca7b80e4216	2021-07-01 18:29:04 UTC

Two Scenario

1. Same hash with different token

1000	0xdc27b766142da84d033bc613331caa233640f8ba9007e2c0129e0ca7b80e4216
1001	0xdc27b766142da84d033bc613331caa233640f8ba9007e2c0129e0ca7b80e4216
1002	0xdc27b766142da84d033bc613331caa233640f8ba9007e2c0129e0ca7b80e4216
1003	0xdc27b766142da84d033bc613331caa233640f8ba9007e2c0129e0ca7b80e4216
1004	0xdc27b766142da84d033bc613331caa233640f8ba9007e2c0129e0ca7b80e4216
1005	0xdc27b766142da84d033bc613331caa233640f8ba9007e2c0129e0ca7b80e4216
1006	0xdc27b766142da84d033bc613331caa233640f8ba9007e2c0129e0ca7b80e4216

0x0000000000000000000000000000000000000000	0xbf59e40112157017b69be5734f8c702dcbd73336	2332601
0x0000000000000000000000000000000000000000	0xbf59e40112157017b69be5734f8c702dcbd73336	2332601
0x0000000000000000000000000000000000000000	0xbf59e40112157017b69be5734f8c702dcbd73336	2332601
0x0000000000000000000000000000000000000000	0xbf59e40112157017b69be5734f8c702dcbd73336	2332601
0x0000000000000000000000000000000000000000	0xbf59e40112157017b69be5734f8c702dcbd73336	2332601
0x0000000000000000000000000000000000000000	0xbf59e40112157017b69be5734f8c702dcbd73336	2332601
0x0000000000000000000000000000000000000000	0xbf59e40112157017b69be5734f8c702dcbd73336	2332601
0x0000000000000000000000000000000000000000	0xbf59e40112157017b69be5734f8c702dcbd73336	2332601

In this scenario, token was transferred from null address, which means new token was minted. Since it has the same hash and the gas fee was paid in one time.

I think when we calculate gas for this kind of token we should use:

receipt_gas_used / number of different token_id within same hash

2. Same token with different hash

10	0x00c1f708e2826aad71beac79c8b777d96aed6a0aef842d69d46bd00050ccf422	2021-07-01 10:31:24 UTC
10	0xab3a528d28098cbd3267d54dc5e94687146bbd1935babc920c739e3d4fdf7df6	2021-07-29 10:34:06 UTC

0x0000000000000000000000000000000000000000	0x84bf627c6c0d04fb8b2bca0e644352fb9e51bfb2	156325
0x84bf627c6c0d04fb8b2bca0e644352fb9e51bfb2	0x52eeb50739502779246cbdfc794eba888b84c04b	76363

In this scenario, the same token has been transferred in different time and result in different receipt_gas_used. The two transactions are independent to each other.

I think when we calculate gas for this kind of token we should use:

avg(receipt_gas_used) group by token_id

3. Token transfer to erc20

10	0x00c1f708e2826aad71beac79c8b777d96aed6a0aef842d69d46bd00050ccf422	2021-07-01 10:31:24 UTC
10	0xab3a528d28098cbd3267d54dc5e94687146bbd1935babc920c739e3d4fdf7df6	2021-07-29 10:34:06 UTC

Calculate gas for each token

1. aggregate gas resulted from all transactions for each token

```
select distinct token_id, accumulative_gas_fee,
  (accumulative_gas_fee*0.0000036756792144+66.1136)*20/3600 *0.0004 watt_per_tran
from
(
select token_id,
sum(receipt_gas_used) over(partition by token_id) accumulative_gas_fee from t
) t2
order by 1
```

token_id	accumulative_gas_fee	watt_per_tran
0	73539	0.000147519790608328 38
10	232688	0.000148819747655645 14
100	2329801	0.000165949335798640 74
1000	2332601	0.000165972206691530 36
1001	2332601	0.000165972206691530 36
1002	2332601	0.000165972206691530 36
1003	2332601	0.000165972206691530 36
1004	2332601	0.000165972206691530 36
1005	2332601	0.000165972206691530 36
1006	2332601	0.000165972206691530 36
1007	2332601	0.000165972206691530 36
1008	2332601	0.000165972206691530 36
1009	149778	0.000148142524180832 02
101	2404052	0.000166555831037192 82
1010	149778	0.000148142524180832 02