Saving Gas without Rollups

(The Easy Way)

Green
Developer
green@kleros.io

kleros.io

@forereth | @greenlucid

KLEROS

European Commission | Winner of the Blockchains for Social Good Prize Horizon 2020

bpifrance

HOSTED BY THOMSON REUTERS INCUBATOR

Easiest Gas Optimization:

- ??? -

Easiest Gas Optimization:

!! Storage !!

Storage

- Expensive

Storage

- Expensive
- Compressible

Storage

- Expensive
- Compressible
- Can be cheaper on some conditions

Always has been
Wait, it’s all **storage**???

Storage Pricing

- 1 Slot = 32 Bytes (this is a lot!)
- Access overhead within a TX:
  - First access (Cold ❄️) => 2100 gas
  - Further accesses (Hot 🔥) => 100 gas

Design Takeaway #1

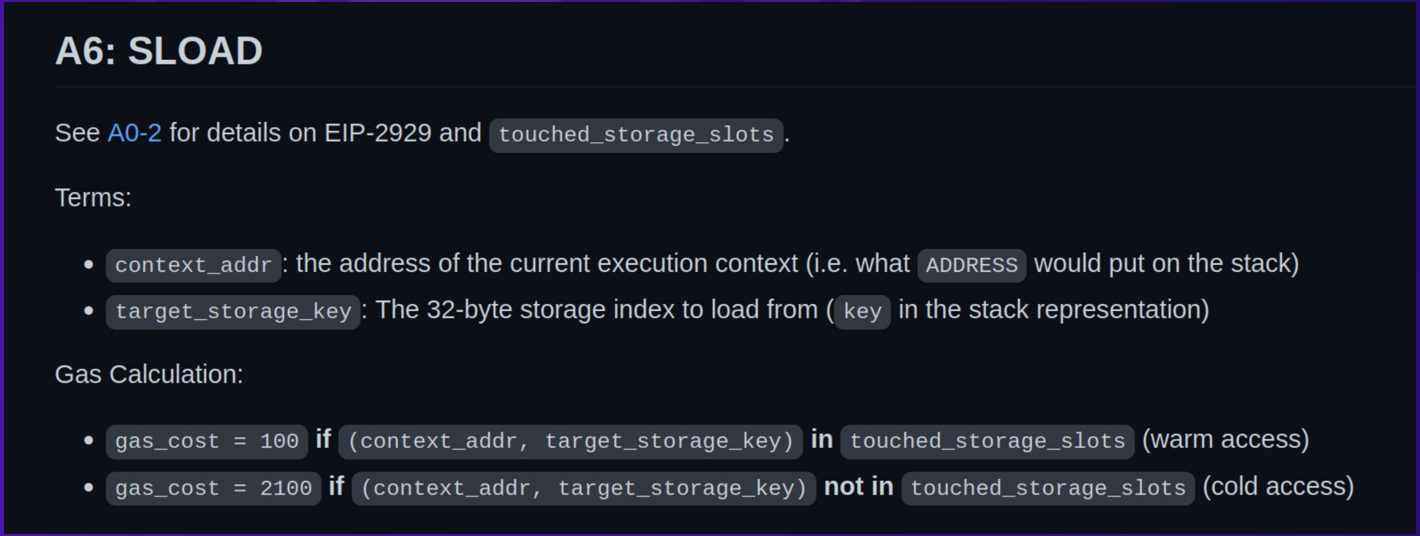
You want to go Hot... 🔥🔥🔥

...not Cold ❄️❄️❄️

Design around minimizing cold accesses from the start.

Reading Storage

SLOAD price = Access price



A6: SLOAD

See A0-2 for details on EIP-2929 and `touched_storage_slots`.

Terms:

- `context_addr`: the address of the current execution context (i.e. what `ADDRESS` would put on the stack)
- `target_storage_key`: The 32-byte storage index to load from (`key` in the stack representation)

Gas Calculation:

- `gas_cost = 100` **if** `(context_addr, target_storage_key)` **in** `touched_storage_slots` (warm access)
- `gas_cost = 2100` **if** `(context_addr, target_storage_key)` **not in** `touched_storage_slots` (cold access)

Writing Storage

A storage slot has three traits:

diff than before tx?	Clean ✨	Dirty ❇️
has something?	Zero □	Non-Zero □
ever accessed in tx?	Cold ❄️	Hot 🔥

Writing Storage

Simple pricing of SSTORE

(Add the cold-hot access price!)

Create state (Clean, 0->!0)	**20_000 gas !!**
Rewrite state (Clean, !0->!0)	**2_900 gas**
Rewrite dirty slot	**0 gas**

Writing Storage

Worth mentioning: Gas refund

(Add the cold-hot access price!)

Delete state (!0->0)	**– 4_800 gas**

Writing Storage

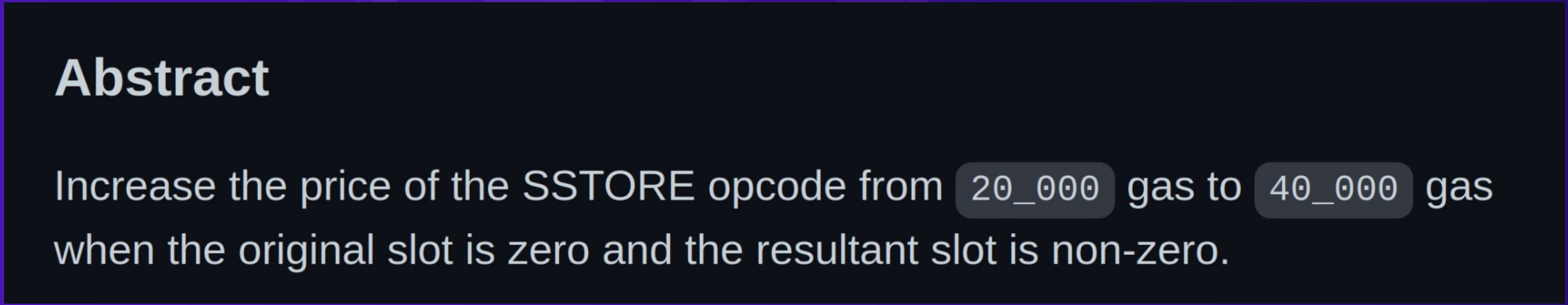
This is not the full story

Want precise prices for every interaction?

Future of storage pricing

Storage may be cheaper than its cost. It may be significantly more expensive in the future.

Example: EIP-5022 (but also, state rent...)



Abstract

Increase the price of the SSTORE opcode from `20_000` gas to `40_000` gas when the original slot is zero and the resultant slot is non-zero.

More gas heavy stuff

Storage is not everything. For didactic purposes, I can’t go into these. Watch out for surprising costs:

- **Logs (emitting events)**
- **Calldata**
- **Address access (e.g. read balance, make call)**
- **CALL* operations**

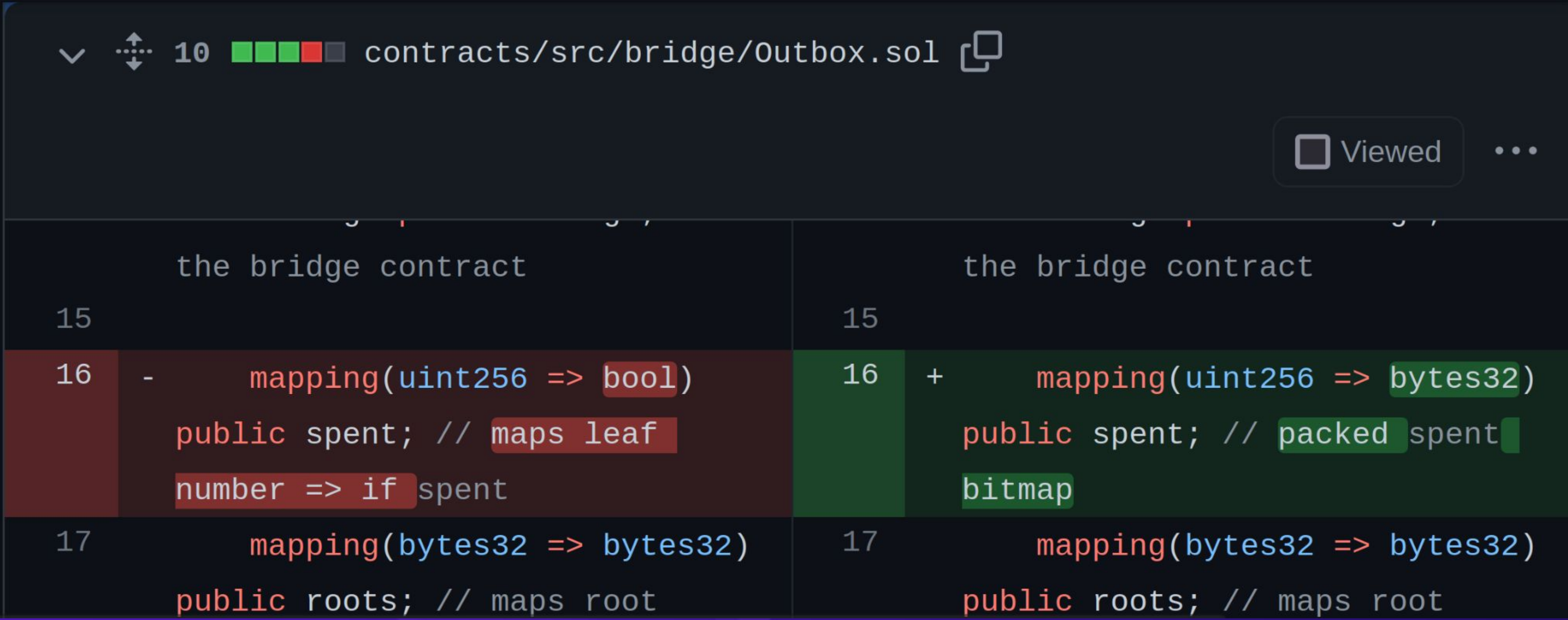
Brief examples of Gas Saves

- Arbitrum Outbox
- UBI token

Arbitrum Outbox

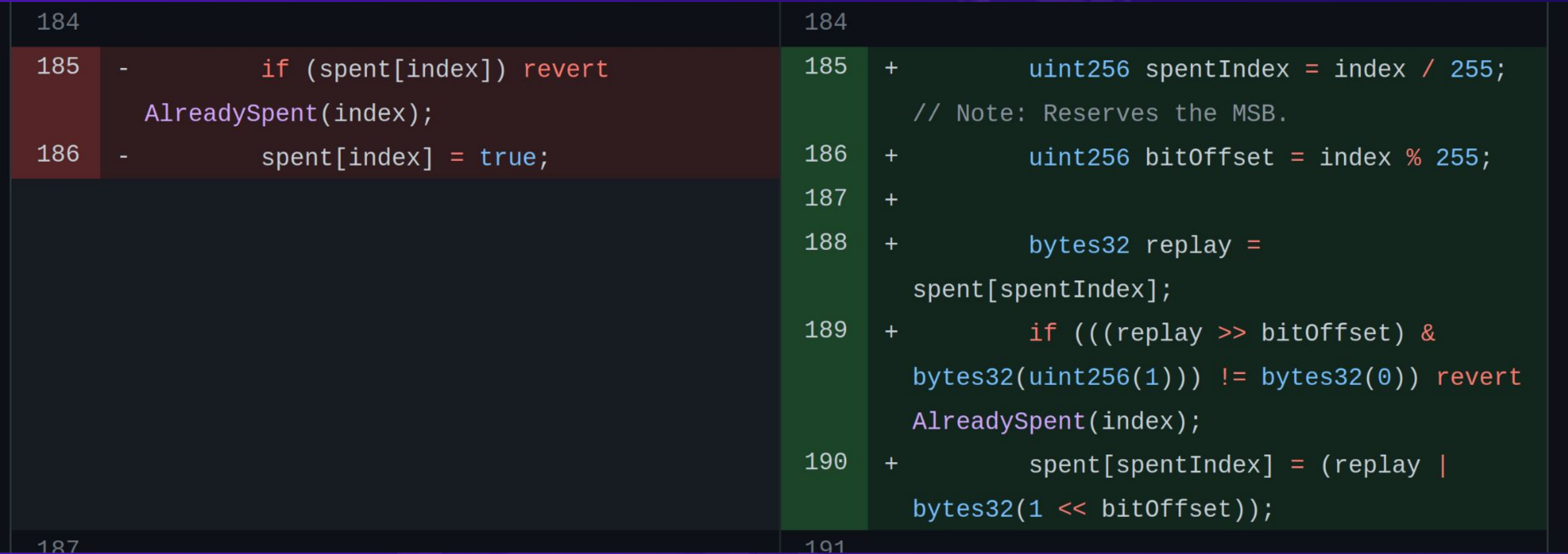
Rollup — — > Mainnet

msg1	//	pending
msg2	//	done
…	//	…



10 🟩🟩🟩🟥▪ contracts/src/bridge/Outbox.sol

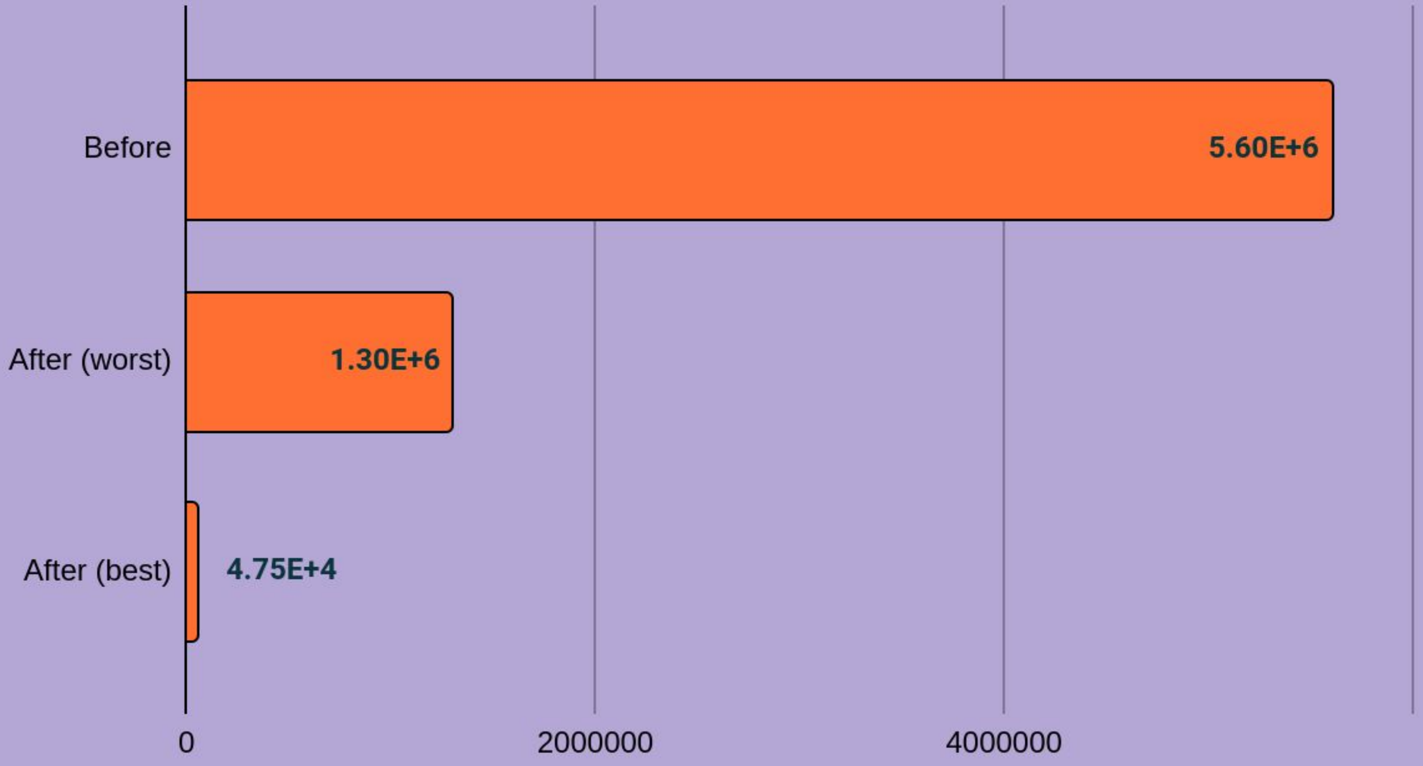
```
     the bridge contract

-    mapping(uint256 => bool) public spent; // maps leaf number => if spent
+    mapping(uint256 => bytes32) public spent; // packed spent bitmap
     mapping(bytes32 => bytes32) public roots; // maps root
```



```diff
-         if (spent[index]) revert AlreadySpent(index);
-         spent[index] = true;
```

```diff
+         uint256 spentIndex = index / 255; // Note: Reserves the MSB.
+         uint256 bitOffset = index % 255;
+
+         bytes32 replay = spent[spentIndex];
+         if (((replay >> bitOffset) & bytes32(uint256(1))) != bytes32(0)) revert AlreadySpent(index);
+         spent[spentIndex] = (replay | bytes32(1 << bitOffset));
```

Storage overhead



Arbitrum Outbox

With ETH @ 1139$ and gas @ 60 gwei
Worst case scenario, save is:

~500$ per batch
(~2$ per msg)

UBI Token

Proof of Humanity

UBI

Vitalik Buterin

Vitalik is the founder of the Ethereum project.

Vouchers

1/1

Last Change:

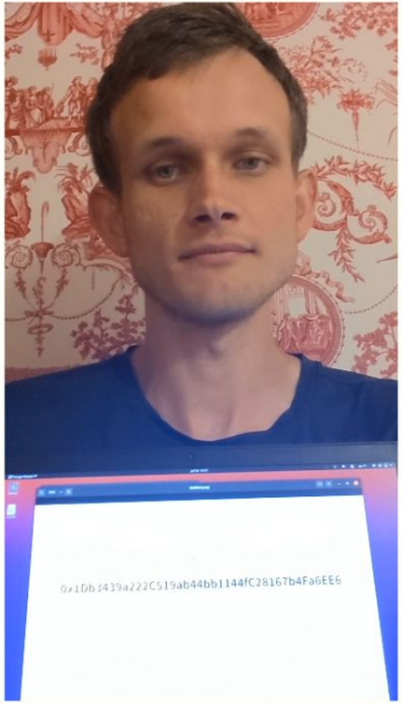
11 months ago

Accepted:

11 months ago

Vitalik Buterin

0x1db3439a222c519ab44bb1144fc28167b4fa6ee6

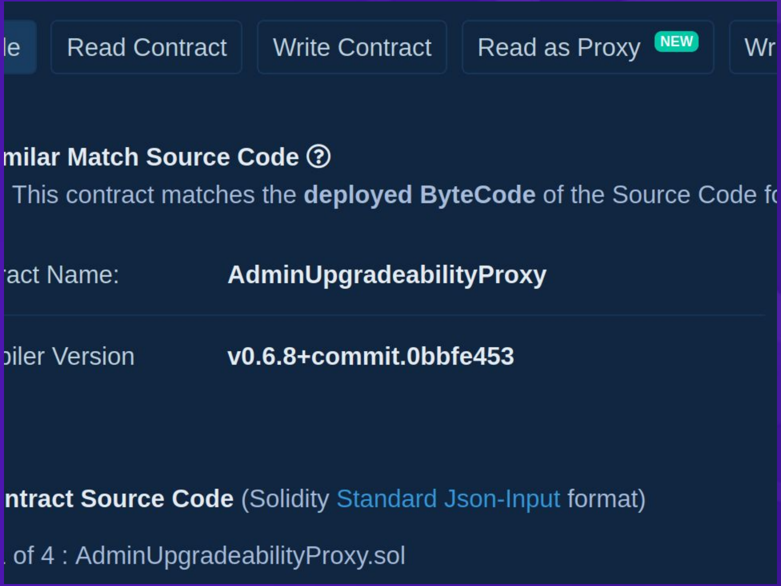


8614.6 UBI

Vouched by:

UBI Token

Upgradeability proxy



milar Match Source Code ?

This contract matches the deployed ByteCode of the Source Code fo

act Name:	AdminUpgradeabilityProxy
piler Version	v0.6.8+commit.0bbfe453

ntract Source Code (Solidity Standard Json-Input format)

. of 4 : AdminUpgradeabilityProxy.sol

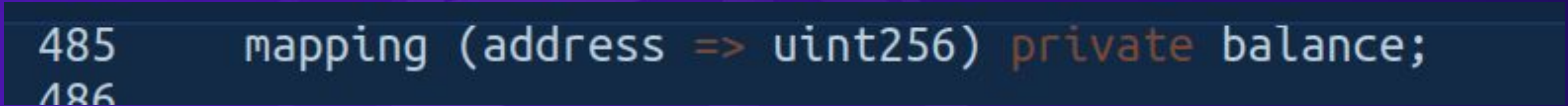
Can’t delve into DELEGATECALL costs.

But:

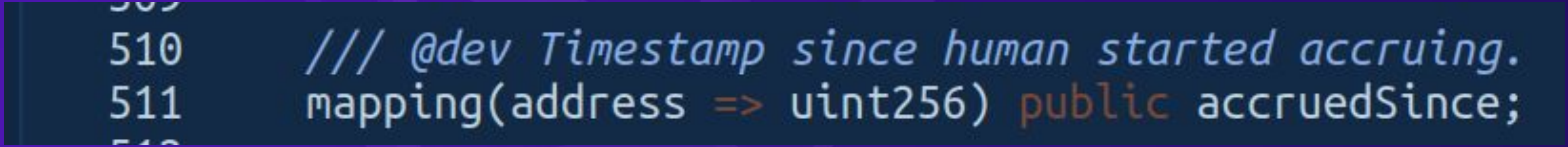
- Addresses have cold-hot:
  - Cold ❄️ address: 2600
  - Hot 🔥 address: 100
- Base cost: 700
- (other stuff)

UBI Token

Scattered information



```
    mapping (address => uint256) private balance;
```

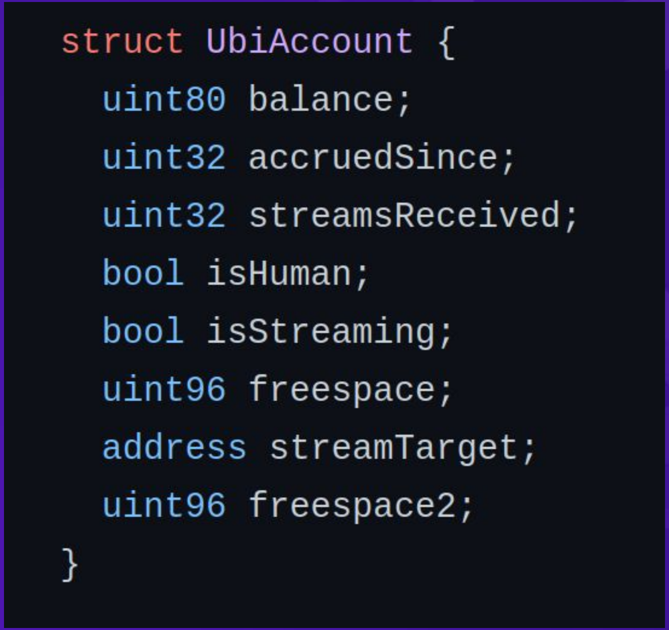


```
    /// @dev Timestamp since human started accruing.
    mapping(address => uint256) public accruedSince;
```

5000 gas to be saved

UBI Token

Packing this data



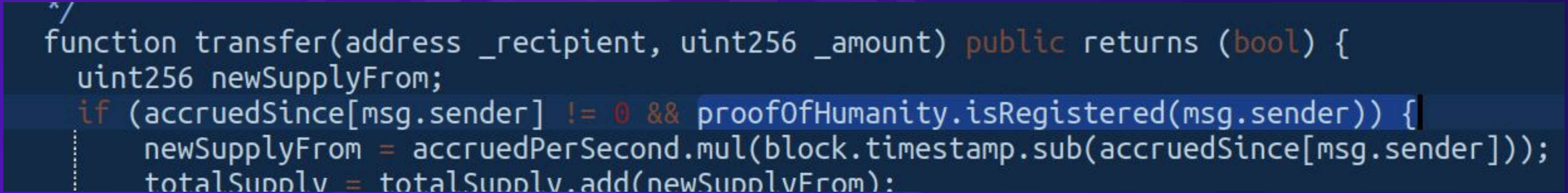
```
struct UbiAccount {
  uint80 balance;
  uint32 accruedSince;
  uint32 streamsReceived;
  bool isHuman;
  bool isStreaming;
  uint96 freespace;
  address streamTarget;
  uint96 freespace2;
}
```

- Only this slot is used on transfer.
- Notice the *isHuman.*

- This is only used when manipulating streams.

UBI Token

Redundant external calls

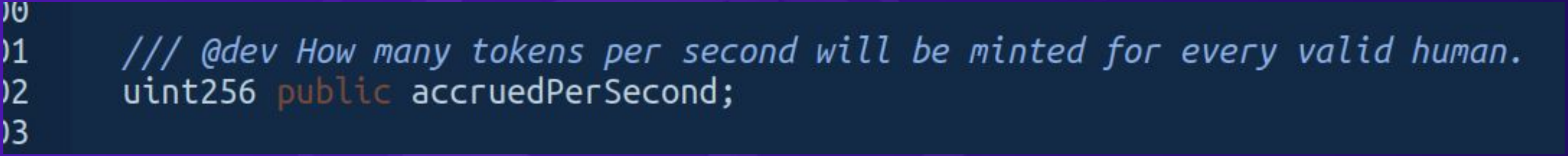


```
*/
function transfer(address _recipient, uint256 _amount) public returns (bool) {
  uint256 newSupplyFrom;
  if (accruedSince[msg.sender] != 0 && proofOfHumanity.isRegistered(msg.sender)) {
      newSupplyFrom = accruedPerSecond.mul(block.timestamp.sub(accruedSince[msg.sender]));
      totalSupply = totalSupply.add(newSupplyFrom);
```

7500 gas can be saved by caching *isHuman.*

UBI Token

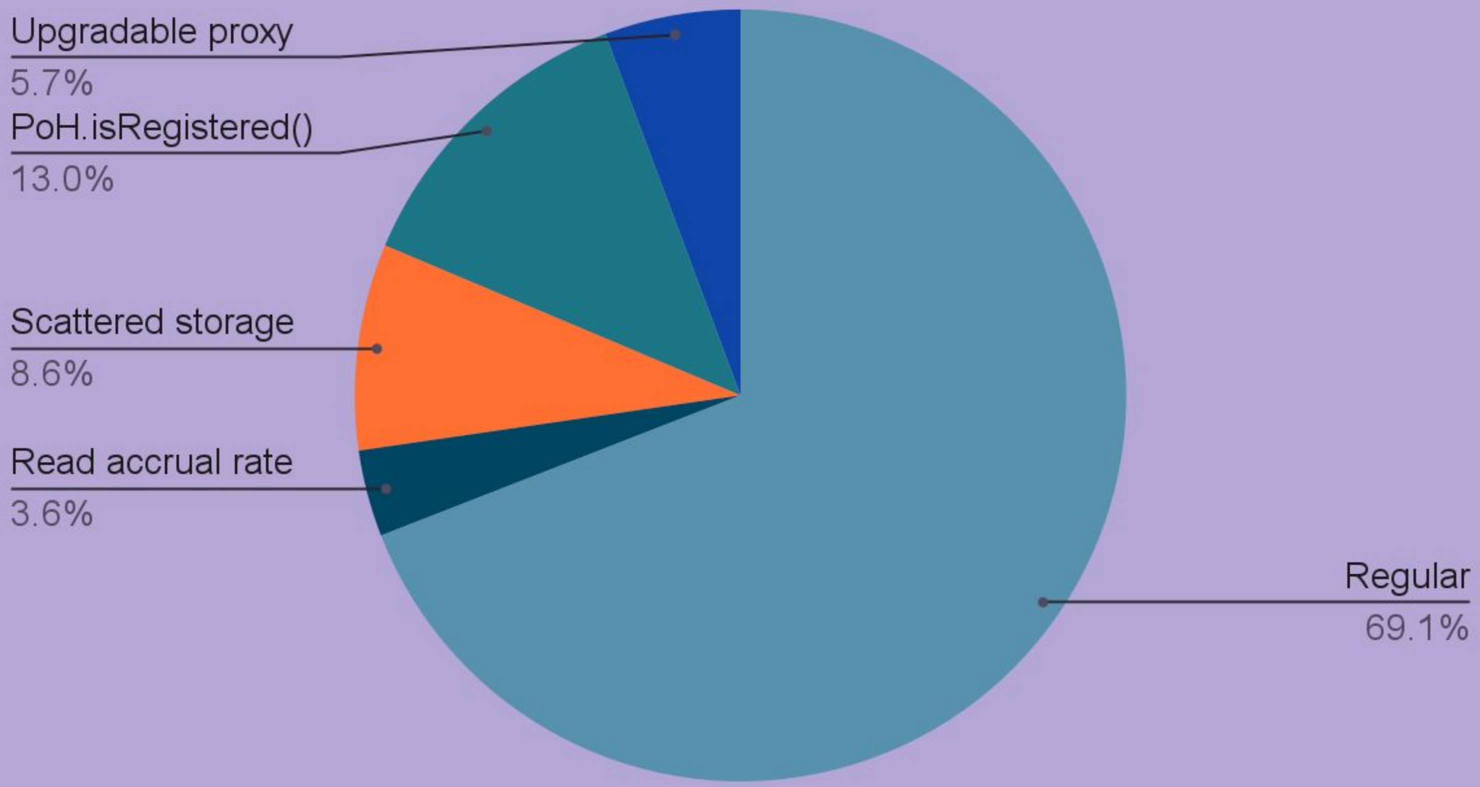
Unneeded storage read



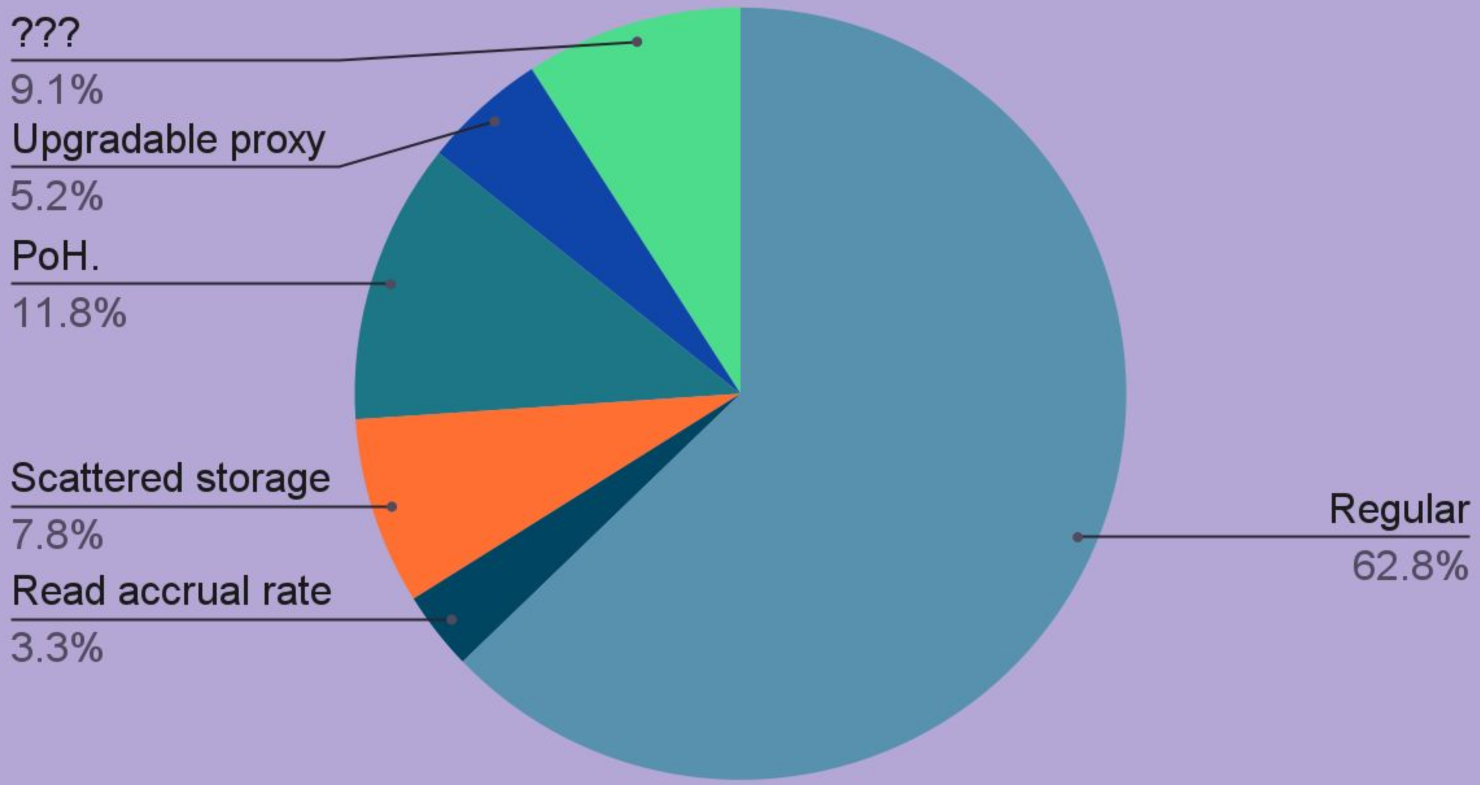
```
    /// @dev How many tokens per second will be minted for every valid human.
    uint256 public accruedPerSecond;
```

Reading this variable on transfer will cost **+2100 gas**.
SLOAD + the stack manipulation needed.
Could just be hardcoded as constant or immutable.

Gas usage per transfer

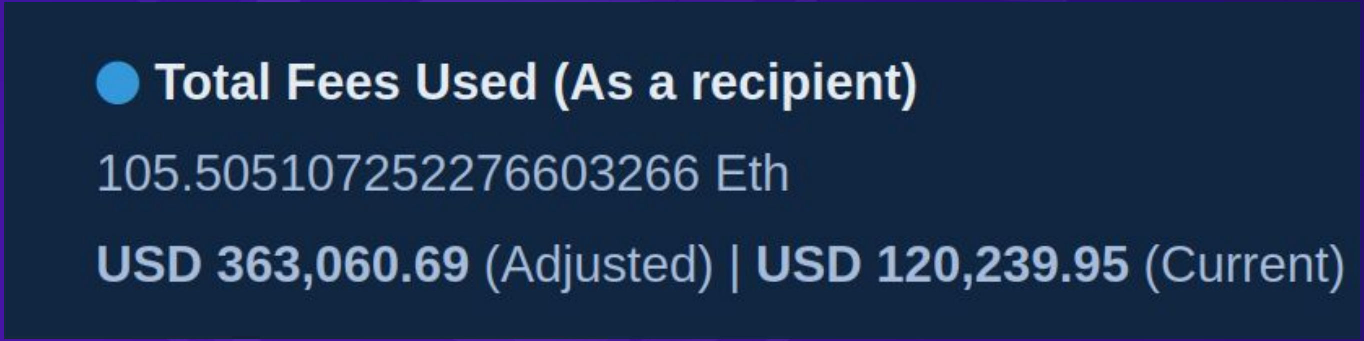


Gas usage per transfer



UBI Token

How much would it have saved?



● **Total Fees Used (As a recipient)**

105.505107252276603266 Eth

USD 363,060.69 (Adjusted) | **USD 120,239.95** (Current)

~136000$

~39 ETH

UBI Token

Other features and improvements

- Precisely obtain totalSupply
  - Apparently, it was giving a lower bound before.

- Streams! You can delegate your UBI stream to another user.
  - It’s ERC-20 right now but I’ll make it ERC-721.

UBI Token

Was it worth to remove upgradability proxy?

- Cheaper
- Safer (immutable)
    - Who can change? Can we trust them?

Takeaways

A few easy optimization patterns

- Reducing stored data
  - Compression ideas
- Reusable storage
- Hot access design
- Deleting state for ephemeral external-id maps

Takeaways

Reducing stored data

Not needed → Not stored

→ Taken care of in design. Especially if not needed on chain

IPFS URIs

Data processing for off-chain consumption

Takeaways

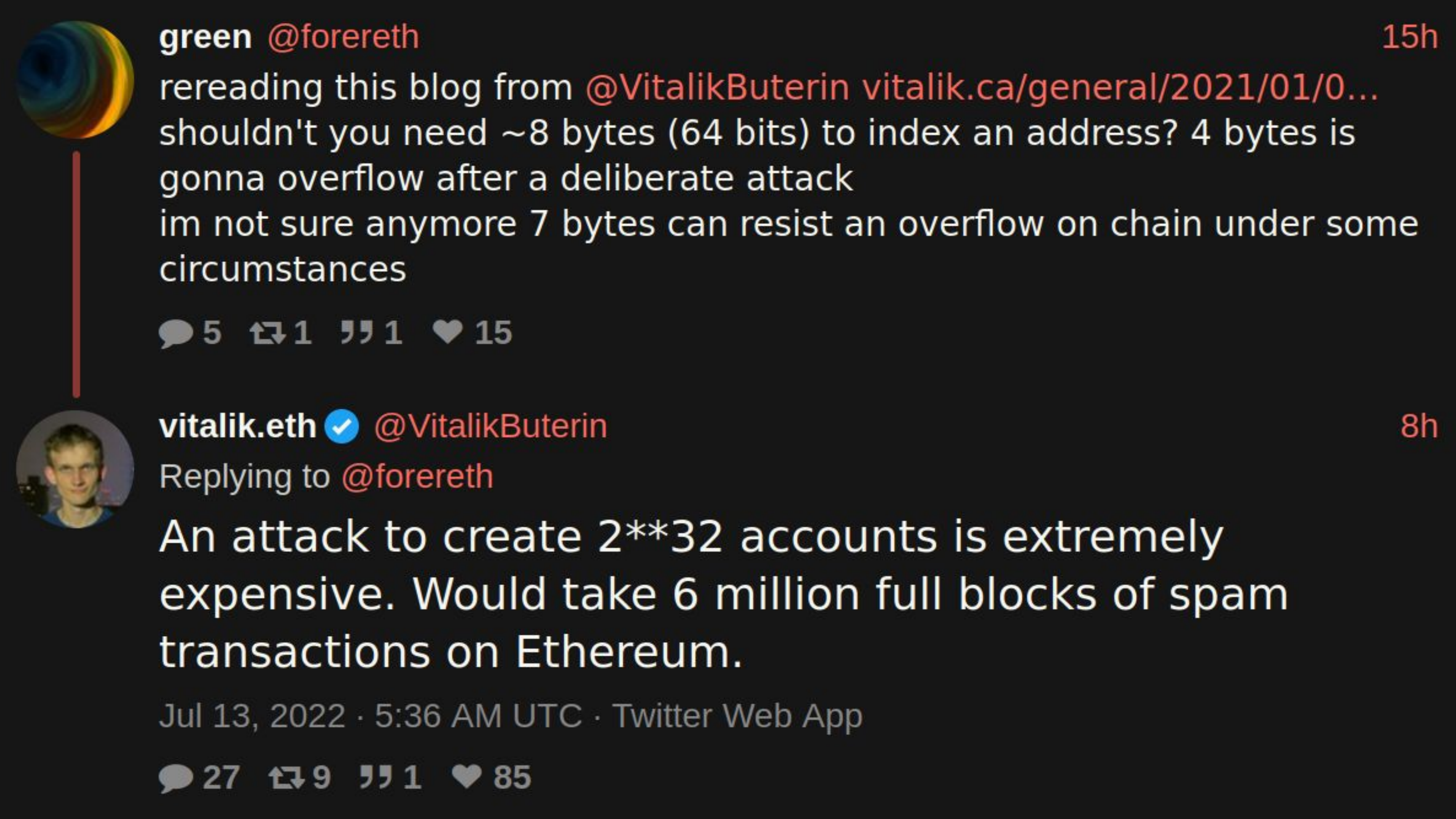
Data packing ideas

Incremental ids

➡ Needs to resist overflow.

- By governor:
  - 8 bits if it’s very infrequent
  - 16 bits to be safe.

- Permissionless: Depends on the cost per increment
  - (sane range: 32 to 64 bits)



green @forereth 15h

rereading this blog from @VitalikButerin vitalik.ca/general/2021/01/0...
shouldn't you need ~8 bytes (64 bits) to index an address? 4 bytes is gonna overflow after a deliberate attack
im not sure anymore 7 bytes can resist an overflow on chain under some circumstances

💬 5 🔁 1 ❞ 1 ♥ 15

vitalik.eth ✓ @VitalikButerin 8h

Replying to @forereth

An attack to create 2**32 accounts is extremely expensive. Would take 6 million full blocks of spam transactions on Ethereum.

Jul 13, 2022 · 5:36 AM UTC · Twitter Web App

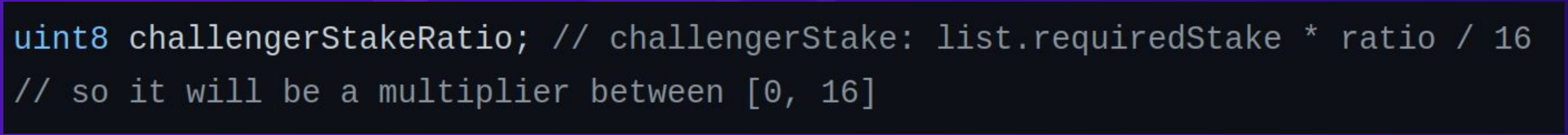
💬 27 🔁 9 ❞ 1 ♥ 85

Takeaways

Data packing ideas

Token amounts

- Can be lossily compressed to 32 bits (also 24 or 16)
- High precision needed? (finance stuff) 64 bits
- Referenced amounts:



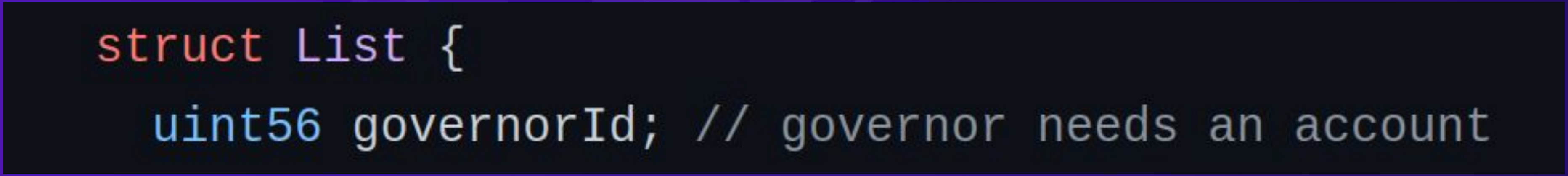
```
uint8 challengerStakeRatio; // challengerStake: list.requiredStake * ratio / 16
// so it will be a multiplier between [0, 16]
```

Takeaways

Data packing ideas

Indexing addresses

- An address is 160 bits, but can be indexed.
- You can do the same for other structs or data types.



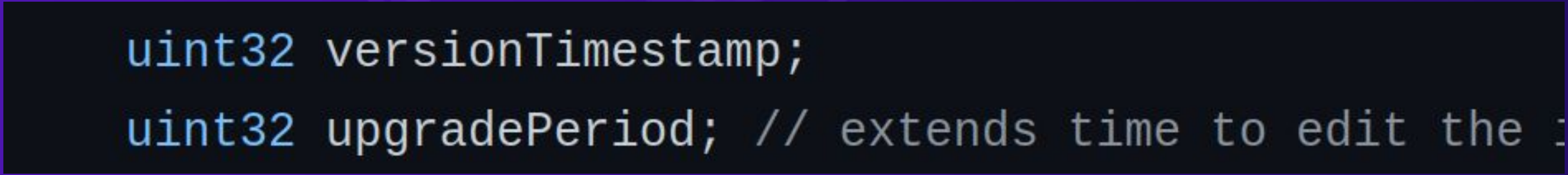
```
struct List {
  uint56 governorId; // governor needs an account
```

Takeaways

Data packing ideas

Time

- 32 bit UNIX second timestamps work until year 2106.
- You can use lower precision with less bits (e.g. days).
- You can (and probably should) hardcode periods.



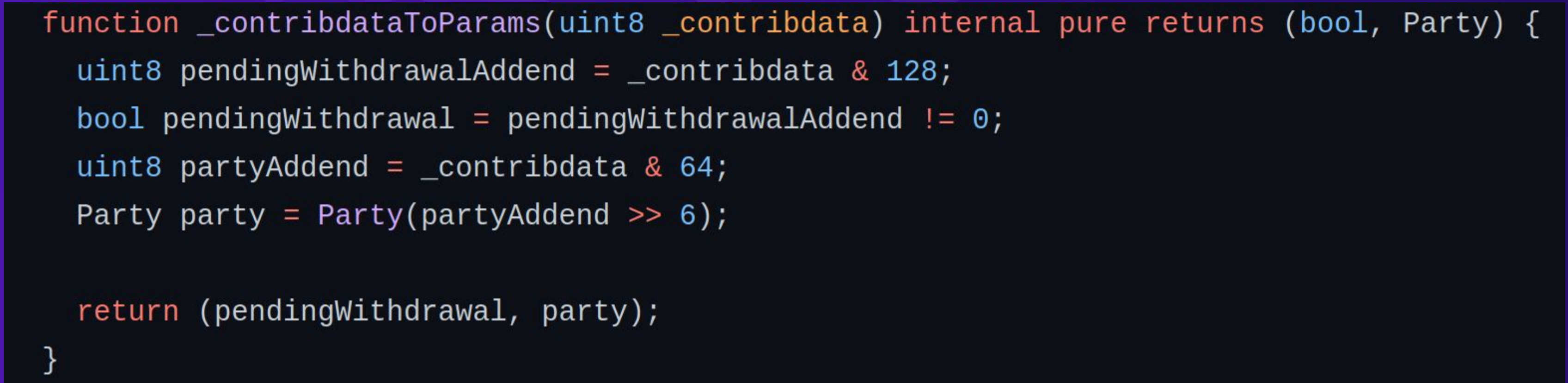
```
uint32 versionTimestamp;
uint32 upgradePeriod; // extends time to edit the
```

Takeaways

Data packing ideas

Enums and booleans

- Consider packing them in uint8s. You can use pure funcs.



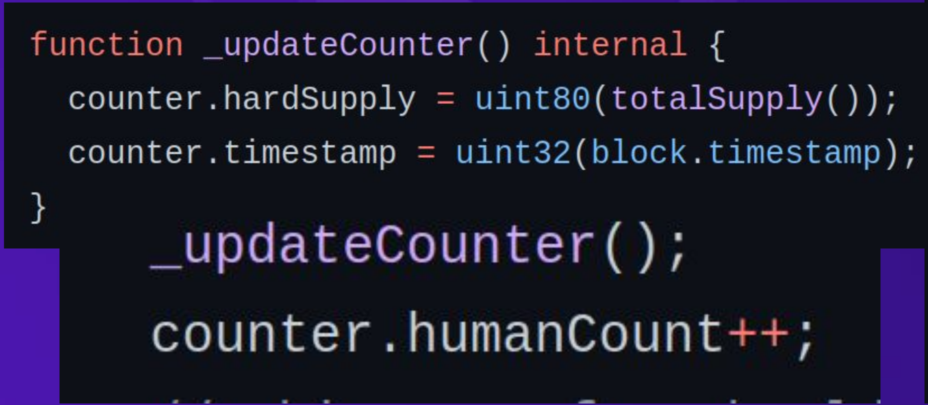
```
function _contribdataToParams(uint8 _contribdata) internal pure returns (bool, Party) {
  uint8 pendingWithdrawalAddend = _contribdata & 128;
  bool pendingWithdrawal = pendingWithdrawalAddend != 0;
  uint8 partyAddend = _contribdata & 64;
  Party party = Party(partyAddend >> 6);

  return (pendingWithdrawal, party);
}
```

Takeaways

Hot access design

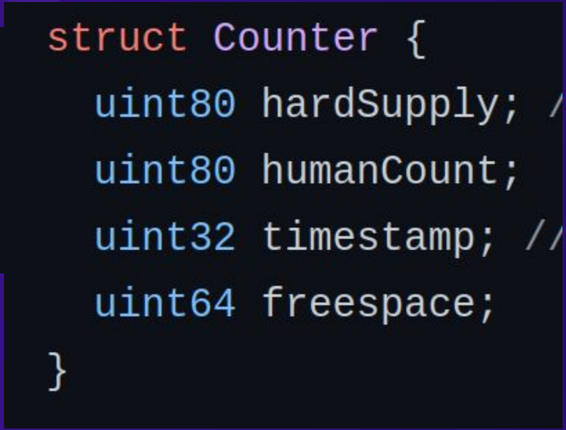
Keep frequently accessed stuff in same slot.

Singletons or global settings too.



```
function _updateCounter() internal {
  counter.hardSupply = uint80(totalSupply());
  counter.timestamp = uint32(block.timestamp);
}
```

```
    _updateCounter();
    counter.humanCount++;
```



```
struct Counter {
  uint80 hardSupply; /
  uint80 humanCount;
  uint32 timestamp; //
  uint64 freespace;
}
```

Thanks!

So much to talk about, but not enough time to cover it all.

Got any questions?

Green
Developer
green@kleros.io

kleros.io

@forereth | @greenlucid

KLEROS

European Commission | Winner of the Blockchains for Social Good Prize Horizon 2020

bpifrance

HOSTED BY THOMSON REUTERS INCUBATOR