Storage Taxes are probably one of the most important scalability solutions after Sharding and Plasma. However, as far as I'm aware there is relatively little development work going into it. I would like to explain the benefits that storage taxes can provide, along with a few solutions to some of the remaining problems it has which can be worked into a future EIP.

Scalability

Much like sharding will make it possible for a participant in the consensus protocol to agree to execute a piece of code in a block without executing the code them self, storage taxes will allow a participant in the consensus protocol to agree that a piece of data is being stored without needing to store it them self. It does this by providing the tools and incentives for users to store the needed data off the chain, and prove its validity whenever necessary. Users will be strongly encouraged to do so by having to pay for storage that they force everyone participating in the consensus protocol to store. Storage taxes will decrease memory requirements for nodes as much as sharding will decrease cpu and bandwidth requirements for nodes.

Micro-POW

Micro-POW is the most exciting opportunity that storage taxes will provide other than scalability.

Transactions require eth to be added to the blockchain. Right now eth is produced when a block is mined in an increment of ~3 or soon 2 eth. This is enough for thousands of transactions to be added to the chain. Unfortunately, all those transactions can be linked to each other because of this eth, undermining other anonymity features until this is solved. Not only that, new users usually cannot mine a block and so if they want to use any dapp, they have to buy eth, usually through a centralized source that often requires ID.

Micro-POW is a simple opcode which takes a target amount of work, and a proof that that amount of work has been completed, and generates an amount of eth proportional to the amount of work doneand increments the callers nonce. Combined with #859

it will allow a brand new account to pay the gas cost for a transaction without being linked to anything else, and without needing prior interactions with the ethereum ecosystem. The sybyl resistance that is usually provided by gas costs will still exist because it will cost more to use this than to pay in eth from mining or staking for anyone executing a large number of transactions.

This addition needs storage taxes to exist to maintain a demand for eth. Currently eth is valuable because it can be spent on transaction fees. With other ways to pay for transaction fees, eth will need a place where it is spent to maintain its value. That can be storage taxes.

Awaken Bloom

One of the most critical features of Storage Taxes is the sleep/awaken function described in more detaihere. Without this, contracts or access to contracts may be lost forever. One issue with sleep/awaken is that awakening a contract requires a proof that the contract has not been awoken since the last time it was put to sleep. This proof is potentially very expensive. The Awaken Bloom is a technique to make this less expensive.

At regular intervals (either every block or some longer interval) in addition to the head of a merkle tree of the awoken blocks being published, a bloom filter of the awoken blocks would also be published. This bloom filter would automatically be available like anything else in the block head, and could easily be used by the awaken opcode to prove that the contract has not been awoken for a large number of blocks for a small per-block gas cost. Bloom filters sometime generate false positive matches, but by including the merkle tree head for each of these, proofs for the false positives can be added the few times it comes up. If this is used, the false positive rate should influence the decision of the time interval used.

Account Creation

One operation that will be even more costly than awakening an account will be creating a new one. Before an account can be interacted with, starting with a nonce of 0, it will need to prove that it has never in the entire blockchain's history been created and then put to sleep. This may not start expensive, but it will grow expensive over time. To eliminate this cost, a new type of public key/private key account can be added. This type of account will be defined both by the private key and the blockhash of the most recent block when it was created. When the account address is determined, before the address is truncated, hash the recent block hash with the public key. This way when you later interact with the address for the first time, the proof will only need to go back to the specified block.

Tax Payment Experience

Users will not want to see their eth balance slowly dropping due to the tax. For this to be possible, users will need to interact

with individual wrapped eth contracts that cannot be awoken without immediately being put back to sleep. To make it possible for the contract to create a limitation like that, the awaken function should instead be an awaken + call function. This will allow wrapped eth contracts to always revert if interacted with by an account that will not immediately put it back to sleep. It will also allow developers to create more complex permissions requirements for keeping a contract awake.