In current ecosystem, there are lot of contracts with the same code but still the code is deployed over and over again for each of the contract. This is a source of redundancy which is not optimal. This post proposes a model to improve upon this issue.

Code and storage for a contract can be separated by treating code as a special kind of account. Any smart contract that needs to be deployed has to have the code already deployed as a special kind of Code

account which doesn't have any storage attached to it. Then a contract can be deployed by pointing to the Code

account.

Rent can be introduced into such a model as follows. When a transaction is sent to a deployed contract, the transaction will pay a royalty for using the code. The royalty collected by the Code

account can be used to pay back the initial deployer and the rest of the royalty is used to pay rent for the Code

account. The royalty that has to be paid, can be set by the contract that uses the Code

. This flexibility allows a developer to decide/adopt the royalty that has to be collected per interaction based on the usage of the contract(ENS kind of contract might have high royalty as they have low code interaction rate). Royalty that has to be paid is specific to a contract rather than code. This may lead to specific contracts always opting for not paying any royalty. So we need a criterion to decide whether a contract has contributed enough to use the Code

. One way to decide this criterion is if the number of interactions with code is proportional to the royalty paid by the contract to Code

. So the criterion can be

Total Royalty paid by contract

= Code Storage Cost

- number of interactions of contract with code

/total number of interactions with code

Assuming that the storage gasprice

doesn't change very often. It should be possible to calculate the code storage rent

for any contract by knowing the block at which contract was deployed and changes to the storage gasprice

after that. Code

and contract can store the number of interactions with each of them. Contract should also store the total royalty it paid.

When a contract performs a self-destruct the royalty that is already paid will not be refunded. The fields stored by a contract like number of interactions and total royalty paid are reset when Code

expires.