dType (Decentralized Type System) on Ethereum 2.0

This discussion is an ongoing effort to research possible implementations for dType on the future Ethereum 2.0.

It is a work in progress. Some ideas might be outdated (especially as Eth2 spec changes), some might be improbable.

Current Ethereum 1.x ERCs:

1. EIP-1900: dType - Decentralized Type System for EVM
2. EIP-2157: dType Storage Extension - Decentralized Type System for EVM
3. EIP-xxxx: dType - Extending the Decentralized Type System for Functions

Prototype and Demos

Current source code, on Ethereum 1: https://github.com/pipeos-one/dType. The Readme.md

page also links to several articles that we wrote on the topic and a playlist of demos.

A demo of dType ver. 1 is dType - Decentralized Type System on Ethereum & Functional Programming

A demo of a file system and a system of permissions, both built using dType types is File System with Semantic Search on dType, Ethereum. Here we also used dType to define ontologies and links from ontology concepts to the filesystem components. We can have a decentralized, browsable Eth2.

dType Intent and Motivation

dType is a Decentralized Type System for a Global OS, for Ethereum. A global type system enables independently built system parts to interoperate. Developers should agree on standardizing common types and a type registry, continuously working on improving them, to become part of the Global OS. Ethereum could become the origin for defining types for other programming languages.

dType can have two parts:

- a global part: a registry of publicly curated types, that benefits most of the projects; this requires consensus.
- an individual part: each project can use its own dType system.

There are other cases where a global scope is needed: anywhere you need fast-access to standardized data. This can, for example, include various ontologies used to label blockchain data or provide transaction metadata (as in the Semantic Search demo).

The global scope is intended for public, highly used data, that would help create a unitary OS across shards and would help off-chain tools to analyze and understand on-chain data. Examples: global ML and AI systems, blockchain explorers that present chain data in rich formats, based on dType types.

dType on Ethereum 2.0

How can such a global scope be implemented in Ethereum 2.0?

From the current information, some ideas are:

A Master Shard (MS)

- will provide data to the global scope
- data inserts in MS - consequence of multiple cross-shard read operations from the original data shard (by other shards)
- it will be a cache of frequently-read data from the other shards, kept in sync with them
- intended for read only, only exceptionally will accept a write; it will operate this write first on the original shard of the data and then, the data will be updated on MS without needing to go through the validator consensus a second time.
- dType can have a custom EE, with only these operations allowed:
- dTypeAdd

- dTypeRemove
- dTypeUpdate
- dTypeGet
- dTypeAdd
- dTypeRemove
- dTypeUpdate
- dTypeGet
- can also hold state roots for shards; similar in concept with a master database (where databases, tables and fields are kept as records)

TBD