Fusotao Greenbook(Draft)

v0.1.2

Fusotao Dev Team

October 18, 2021

1 Introducation

Fusotao is a set of network protocols which are consisted of either a rule of data consistency or some certain constraints of data modification including not only a transfering constraint but also a matching verification rule. The matching verification sub-protocol, a.k.a Proof of Matching, is the main difference from other blockchains.

2 Matching System

Before start introducing Fusotao Protocol, let's take a look at matching system and consider why a matcher's outputs should be verified.
Matching system is a trading platform enables user to pricing orders. The core component of a matching system is a data structure named orderbook which stores all orders according to the price and time, a placed order must follow the sequence to trade if price meets while the owner of an order would be ignored. Usually, to trade on a matching system, users may hande over their account ownership so the matcher can mutate the accounts. In another words, trading on a matcher relies on human trust.
Any data modifications shall obey the rules below:

Rule 1 *Mutable data shall not be shared, and shared data shall be immutable.*

Rule 2 *There shall be only one associated mutator once data shared.*

Compared to the transfering transaction, the matcher as a mutator must modify the orderbook's state for each order rather than just update the states of sender and receiver. So, a matching system is a strict serial system which can be represented by the following procedure:

$$E_i + Orderbook_{i-1} \Rightarrow Orderbook_i + R_i$$

Consider an order placed. Let x be a price, y be an amount, (x_i, y_i) be the i_{th} maker by the order of price and time.

$$mb_i = -y_i$$

$$tb = \sum_i y_i$$

$$mq_i = -x_i \times y_i$$

$$tq = \sum_i x_i \times y_i$$

where

- $matches(x, x_i)$ is $true$

Blockchain is another typical serial system whose key rule is determining the order of accepted incoming events, e.g. the Bitcoin network uses PoW algorithm and the Longest Chain rule to ensure the unique sequence of transactions. It seems nature to build a matching system on blockchain just like a plain transfer function. But unfortunately, due to the limitation of block capacity and high latency, it is not straightfoward to do it.
An order must occupy 73 bytes at least:

$$ID + Owner + Price + Unfilled + Direction = 73bytes$$

where:

- $Price$ and $Unfilled$ are 128-bits fixed number
- ID indicates the order
- $Owner$ is the pubkey of user
- $Direction$ is 1-bit direction

Imagine there are tens of thousands of orders in a single orderbook, it is not acceptable to store all orders in the blockchain state machine. But only keeping some essential data to validate the matching results is possible. In the next section, we will introduce how to validate the matching results without holding the whole orderbook.

3 Global States

Sparse Merkle Tree is a full binary hash tree with fixed-depth which can be used for checking whether a node belongs to the tree. A node of Sparse Merkle Tree can be represented by the following function:

$$\psi_x = \lambda(\psi_{xL}, \psi_{xR}),\ height \neq 0$$

$$\psi_x = v,\ height = 0$$

where

- x is the key of a node calculated by $\lambda(value) >> height$, e.g. $0x00...a82e$

The capacity of a Sparse Merkle Tree depends on the hash function of the tree, e.g. a Sparse Merkle Tree using Sha256 has 2^{256} leaf nodes with height=256(root excluded). Given a data v, we can simply verify whether it belongs to a certain tree by calculating $height - 1$ times hash function:

```
let h = hash(v)
from 0 to 255:
    do h = hash(h, sibling_of_h)
return h == root
```

It is quite simple for validators, but not good for provers. Storing all 2^{257} (intermediate nodes included) nodes is unpractical for any storage system. Consider the distribution of a real Sparse Merkle Tree, most of the leaf nodes are empty, so are the itermediate nodes, that's why we call it sparse. In an empty plain Sparse Merkle Tree, a node at height h has a certain value:

$$\psi_h = \lambda^h(\phi, \phi)$$

To avoid precalculate hash for each height, we can redefine the hash function like below:

$$\lambda(\phi, \phi) = \phi$$

$$\lambda(\alpha, \phi) = \lambda(\phi, \alpha)$$

For data updates, once a leaf node inserted, the nodes along the path would be updated until root. Since we have the first optimization, we can infer that there will be a mount of redundant nodes along the path which can be omitted. e.g. Inserting a single node with key=0xff..ff, value=v into an empty Sparse Merkle Tree, the parent node key=0x7f..ff would be value=v either, so are the rest of the nodes along the path. Back to the matching procedure, since we've done an optimized Sparse Merkle Tree so can encode the orderbook into it and only store the root hash on chain and leave the whole tree to matchers. Once a matcher executes the i_{th} event off chain and submits the makers at $i-1_{th}$, we can simply validate the results like above.
Fusotao defines 2 types of key as leaf keys of Sparse Merkle Tree:

$$orderbook(symbol) = \lambda(1, symbol)$$

$$account(currency) = \lambda(0, currency)$$

where

- $currency$ is a 32-bits unsigned number represented currency
- $symbol$ is consisted of $base$ currency and $quote$ currency

There are no keys for encoding orders, instead, the sum of the orderbook's size is enough for validators except the condition that matcher doesn't pick the best makers. In the near future, we may add best price to each orderbook node to get rid of this.
A matcher has a specific global state at the i_{th} event which can be verified by the leaf values:

$$orderbook_i = ask_i << 128 + bid_i$$

$$account_i = available_i << 128 + freezed_i$$

where

- all numerics are 128-bits represented unsigned fixed number with $scale$=18

4 Proof of Matching

Given a user-signed event E_i and some merkle paths $P_{(i-1,j)}$ at $i-1$, a validator can verify it using matching procedure with well-known root hash S_{i-1} stored on chain:

$$\sum_{j=0}^{n} belongs(P_{(i-1,j)}, S_{i-1}) = n$$
$$P_{(i-1,j)} + E_i \Rightarrow S_i + P_{(i,j)}$$
$$\sum_{j=0}^{n} belongs(P_{(i,j)}, S_i) = n$$

A matcher must maintain a Sparse Merkle Tree and update it every time after event applied. Let (x, y) be an ask-limit order's price and amount of symbol (b/q) as i_{th} event, (x_j, y_j) be the j_{th} maker exists at $i-1$, then the proof generated by matcher would be(fee not included):

$$account(b)_{(i,j)} = account(b)_{(i-1,j)} + y_j << 128$$

$$account(q)_{(i,j)} = account(q)_{(i-1,j)} - x_j \times y_j$$

$$account(b)_i = account(b)_{i-1} - \sum_j y_j << 128 + y - \sum_j y_j$$

$$account(q)_i = account(q)_{i-1} + \sum_j x_j \times y_j << 128$$

$$orderbook(b,q)_i = orderbook(b,q)_{i-1} + (y - \sum_j y_j) << 128 + orderbook(b,q)_{i-1} - \sum_j y_j$$

Once verified, the accounts can be mutated on chain and update root hash to S_i.

5 Substrate Implementation & Cross-Chain

As an abstract protocol, PoM can either be built on existing blockchains as a series of contracts or run as an independent network. Consider the gas fee and latency, we decide to implement Fusotao protocol by substrate as a permissionless network which permit submitting proofs with zero-cost and low-latency. In this scenario, Fusotao network acts a decentralized infrastructure rather than a specific DEX. Thus, users may not handle over their accounts ownership to a matcher but just authorize the mutation rights.

Substrate is a complete blockchain framework with a set of built-in tools including LevelDB storage, libp2p communication, etc. Especially, the cryptography of Substrate is fully retained in Fusotao runtime which makes it much reliable. In another word, the Sr25519(Schnorrkel/Ristretto x25519) keys for user singature and Ed25519 keys for node signature used in other Substrate-based networks can be recognized in Fusotao network with changing the first type byte of ss58check address to wildcard 42. The acceptable Fusotao ss58check address format is shown below:

$$address = base58(type + pubkey + checksum)$$

The Fusotao runtime provides a set of core APIs abount PoM:

- $claim$: Claim as a matcher and prover by staking 1% TAOs of current issurance.
- $prove$: Submit the original user-signed order and associated proof.
- $grant$: Authorize a matcher some tokens without transfering ownership.
- $revoke$: Revoke authorization from a matcher.

As an isolated network, it is necessay to bring existing assets to Fusotao. Inspired by NEAR Rainbow Bridge, Fusotao provides a trustless component bridging to NEAR mainnet called Avatar Messenger. Avatar Messenger is consisted of an off-chain relayer, a built-in pallet in Fusotao runtime as a light client of NEAR and a ownerless contract deployed on NEAR network as a light client of Fusotao. Fortunately, NEAR and Substrate both support Ed25519 signed block, bridging to NEAR is much easier than bridging to Ethereum. Because a finalized block will always be within the canonical chain, which means both clients needn't worry about chain fork and the relayer doesn't have to submit all blocks.