ION Finance:
Advanced Dex for The Open Network

Version 0.2

March 2024

Authors

ION Finance Team

1. Introduction

In the current decentralized exchange landscape, two dominant models have emerged: Central Limit Order Book (CLOB) exchanges and Automated Market Makers (AMMs). Each model presents specific advantages and limitations affecting price precision, discovery, and liquidity.

CLOBs allow traders to place orders at specific prices, effectively reducing issues related to price slippage commonly observed in AMM models. The ability to set exact prices aids in the more effective utilization of price discovery mechanisms. However, In scenarios of market illiquidity, the CLOB model encounters difficulties in executing trades without significant price impact.

To address liquidity constraints inherent to CLOBs, ION introduced **Liquidity Providers** as a supplement. These entities contribute liquidity to the trading pool and, in return, accrue transaction fees.

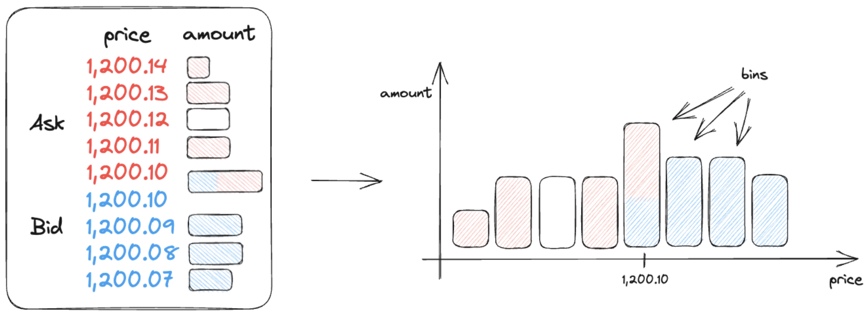
Advantages of the Liquidity Provider Model:

- Abundant Liquidity: A constant influx of liquidity facilitates the execution of trades in less liquid market conditions.
- Concentrated Liquidity: Liquidity Providers tend to allocate liquidity where trading volume is most concentrated, optimizing liquidity availability.
- Reduced Slippage: Enhanced liquidity pools contribute to a reduction in price slippage during trade execution.

This paper introduces an innovative feature developed by ION Finance to leverage the strengths of both CLOB and AMM models: the 'Bin.' This concept combines the advantages of price precision and effective price discovery from CLOBs with the enhanced liquidity aspects of AMMs. The 'Bin' offers a unique trading experience by mitigating the limitations inherent to each existing model.

2. Design

2.1 Bin Introduction



In the order book, orders are categorized into Bid and Ask orders. Each order has a price and an amount value, and the price is separated into minimal ranges.
The amount corresponds to liquidity. A range with a large amount is said to be high liquidity, while a tick with a small amount is said to be low liquidity.

In ION Finance, liquidity can be represented by a graph with price and amount as the two axes, corresponding to the order book in the figure on the left. The Liquidity Bin is the smallest division unit, meaning that a bin represents liquidity in a minimal price range.

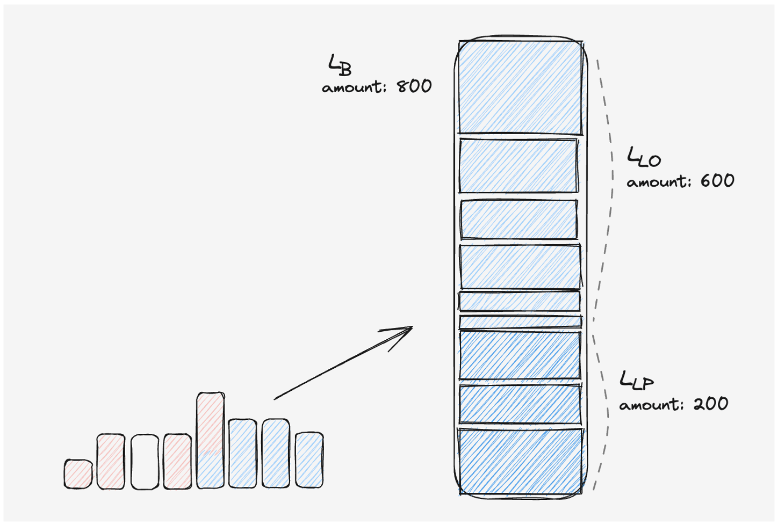
2.2 Bin Composition

The Liquidity Bin is a pivotal construct for implementing our decentralized exchange. It serves as a mini reserve pool with a fixed price. ION takes the traditional order book model found in centralized exchanges and adapts it to fit the decentralized setting by involving two primary participants:

- Makers: Individuals who create limit orders on the order book. Their purpose is to set a specific price for exchanging tokens.
- Liquidity Providers: Individuals who provide liquidity to earn transaction fees.

It's worth noting that our DEX doesn't support taker orders. Instead, orders are filled through instant swaps or market orders. This unique mechanism is made possible due to the nature of the active bin, which allows for more immediate and straightforward transactions.

As for the liquidity in each bin, it's sourced from both makers and liquidity providers, combining to offer a dynamic and robust liquidity pool:



For liquidity management, each bin (denoted by L_B) sources liquidity from limit orders(L_{LO}) and liquidity providers(L_{LP}) Therefore, the liquidity of each bin can be naturally expressed like below.

$$L_B = \sum L_{LO} + \sum L_{LP}$$

Users can utilize all the liquidities in L_B, but when dealing with other bin properties, such as transaction fees, we need to use L_{LP}

And the total liquidity of a trading pair(L_T) is simply the sum of the liquidity across all bins.

$$L_T = \sum L_B$$

2.3 Bin structure

Within each bin, the price gradient(p) remains constant However, unlike other AMMs' constant product formula, the price is decoupled from the reserve states of X and Y. This means that knowing the price(p) and the amount of x (or y), one cannot determine y (or x).

The liquidity in each bin adheres to the constant sum price invariant:

$$Px + y = L_{LP}$$

Where:

- x is the number of assets X provided by liquidity providers.
- y is the amount of assets Y provided by liquidity providers.
- L_{LP} is the bin's liquidity provided by liquidity providers.
- P is the price defined by $P = \frac{\Delta y}{\Delta x}$

2.4 Price Coverage

To ensure that our DEX accommodates a wide range of trading activities and asset prices, it is crucial to calculate the maximum number of bins the system can support. The price of the quote token for the base token offered by ION Finance spans from 2^{-128} to 2^{128}.

Given that bins need to cover this extensive price range while also providing appropriate spacing for trading participants, we use the following equation to calculate the maximum number of bins a pool can have:

$$1.0001^{x} = 2^{128}$$

$$x \simeq 8.87272 \times 10^{5} = 887,272$$

Therefore, the maximum number of order books for both buy and sell directions is $887,272 \times 2 = 1,774,544$

With the maximum number of bins calculated, the ID of a bin can be efficiently expressed within a range of up to $2^{21} = 2,097,152$

While maximizing the number of bins for better market granularity is tempting, doing so has implications for storage costs. More bins mean more data to store, which increases the cost of maintaining the smart contracts. Since liquidity providers and traders bear these costs, pool creators must balance granularity with cost efficiency.

2.5 Market Aggregation

A unique feature of our DEX is that the constant sum curve can intercept both the x and y axes, allowing reserves of X or Y to be depleted. When this occurs, the current price shifts to the adjacent bin.

In any market, only one bin can contain reserves of both X and Y - termed the ***active bin***. Bins to the right of the active bin will contain only X, and those to the left will only contain Y.

For instance, in the TON/USDC pool, set Y as USDC and X as TON. If the active bin is 100 TON, bins to the left will contain only USDC, and those to the right will contain only TON.

Swaps may span multiple bins. Starting from the active bin, the swap will consume bin liquidity until the desired amount is met or the bin is emptied. If a bin empties, the next closest bin's liquidity is used, making it the new active bin.

3. User Actions

3.1 Provide Liquidity

Liquidity is only available in the active bin. Lp providers provide liquidity to earn trading fees.

Invariant

$$Px + y = L_{LP}$$

Suppose a liquidity provider wants to add Δx and Δy to the pool. After the liquidity is added, the new invariant will be:

$$L_{B,new} = P(x + \Delta x) + (y + \Delta y)$$

To calculate the percentage of the total pool that the LP will own after providing Δx and Δy, we can use:

$$LP\ Share\ \% = \frac{P\Delta x + \Delta y}{L_{B,\,new}}$$

3.2 Limit Order

Limit orders can be applied to bins that have the same liquidity as the token you want to apply the order for and not on an active bin.

In a single bin, the liquidity is:

$$L_B = \sum L_D + \sum L_{LP}$$

Limit order increases the liquidity of the bin.

For amount of limit order L_{LO}, the liquidity after limit order is:

$$L_{B\_new} = L_{B\_old} + L_{LO}$$

When trading market order, the liquidity set as limit order will be traded first.

3.3 Market Order

Within a bin, the overall liquidity(L) and the price (P) remain constant as exchanges occur. Only the composition of assets x and y changes. The exchange ratio for the two tokens within the bin can be described by:

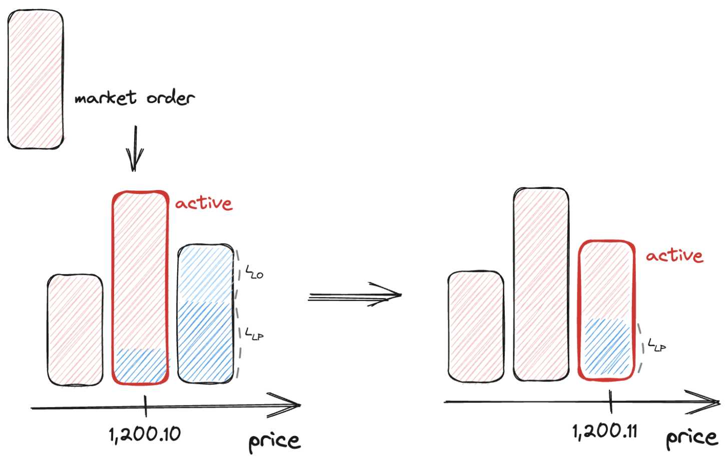
$$y =- Px$$

Given this, the maximum amount of assets x or y that can be exchanged within the current active bin can be calculated as follows:

$$x_{max\_exchangeable\_in\_activet\_bin} = x_{current\_reserve}$$

$$y_{max\_exchangeable\_in\_activet\_bin} = y_{current\_reserve}$$

The illustration below shows the process of market order.



When processing market orders, in a bin, the liquidity set by limit orders will be traded first. If that liquidity is exhausted, the liquidity provided by liquidity providers will be traded.

If the exchange amount surpasses the maximum exchangeable quantities calculated above, the swap may span multiple bins and change the active bin.

To generalize this, let i be the ID of the currently active bin, P_i its price, and L_i its liquidity. if a user inputs an amount x_{input}, the amount of y they can receive (Δy) would be:

$$\Delta y = (\frac{L_i}{P_i} - x_i) + \sum_{k=i+1}^{i_{max}} (\frac{L_k}{P_k})$$

Here, $\frac{L_i}{P_i} - x_i$ represents the maximum amount of y that can be exchanged in the currently active bin i. The sum term adds up the maximum amount of y that could be obtained from subsequent bins, starting from i to i_{max}.

4. Architecture

4.1 Avoiding Unbounded Data Structures

In TEP-74, balances per user address are subdivided into discrete contracts to facilitate storage capacity expansion indefinitely. This approach to distributed storage offers the dual benefits of reducing transaction gas costs and harnessing the power of parallel processing. Considering these advantages, instead of consolidating multiple bins within a single contract, each bin is represented as an autonomous contract.
In ION finance, bins are managed as contracts. Like other TVM contracts, the address of a bin is generated deterministically. The arguments are the address of the token pair and the binstep, which represents the price ratio.

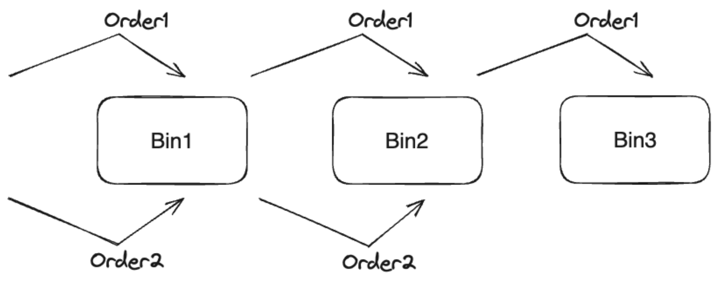
$$bin\_address = f(token\_pair\_address,\ bin\_step)$$

4.2 Order Queue

In specific scenarios, the synchronous processing of multiple bins may be unavoidable. Since bins are represented as separate contracts, this architectural setup entails the execution of transactions in parallel. To manage such synchronous orders seamlessly, we utilize Order Queues.

4.3 Concurrency Conflict

In parallel execution, the simultaneous arrival of multiple orders can result in subsequent orders encountering failures. For instance, in the scenario where two market orders are submitted concurrently, there exists a significant likelihood that the later order may encounter failure if the preceding order modifies the active bin.

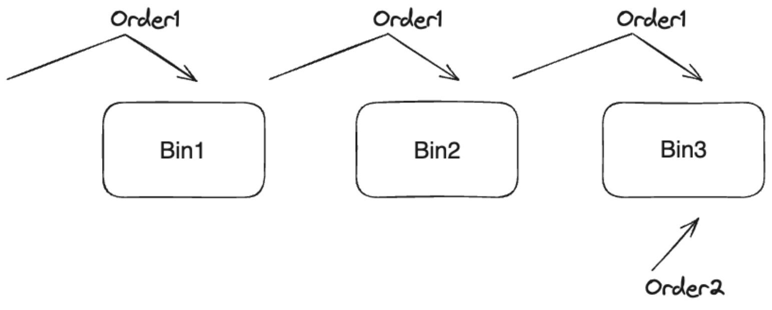


In a scenario where two users concurrently attempt to execute market orders, it is worth noting that market orders commence with the current active bin. When the liquidity of a particular token within the active bin is depleted, the system proceeds to identify the next active bin, repeating the same process. To provide a more precise illustration, let us consider a situation where users A and B simultaneously submit Order1 and Order2, respectively. Order1 initiates its execution from Bin1, depleting the liquidity in Bin1 and Bin2, and then proceeds to Bin3. In contrast, Order2 fails due to unavailable liquidity in Bin1 and Bin2.

The issue described above arises due to the asynchronous nature of Order1's execution and the update of the active bin ID. Consequently, the active bin ID acquired by User B's trade becomes outdated from the bin's perspective. As a result, User B's order traverses the messaging system and eventually encounters failure. In situations with a sudden surge in orders within a liquidity-constrained protocol, this scenario can lead to the failure of numerous orders.

4.4 Utilizing a Queue Mechanism

Orders are queued to resolve the above issue, meaning that no other market orders will be executed if a market order is executed on a pair. If there is an executing order, it will be queued. As a result, after Order1 finishes, Order2 will execute orders from Bin3.



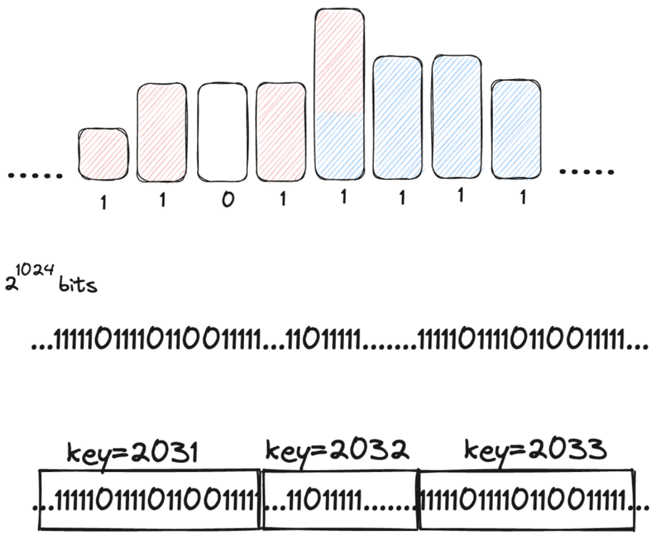
The queue system uses TVM's dictionary, wherein timestamps are the keys, and the corresponding values encapsulate order-related information. A simulation was conducted on the TON testnet to evaluate the processing duration required for a batch of queued orders. This simulation encompassed 100 orders, each of which induced 3 bin changes. The simulation results demonstrated that all transactions were expeditiously processed within just 2 blocks.

The code utilized in the simulation is publicly accessible on GitHub. Within the simulation, four counters are present, one acting as a route contract (responsible for receiving user messages), and the remaining three serving as bin contracts. These counters fulfill essential roles. Specifically, the first function involves incrementing the stored counter and dispatching a response message to the originating address. The second function entails the transmission of a message to the bin. When all three bins receive the message, the corresponding order from the Order Queue is removed. You can explore an illustrative example of processing 100 orders through Tonviewer.

Notably, TON is designed to handle an impressive throughput of over 1 million transactions per second, with a tendency for orders to be executed promptly without frequent queuing.

4.5 Liquidity Tracking Bits

When a bin's liquidity is depleted during order processing, the subsequent bin must be determined. While one approach involves incrementing the bin ID by one, this method is not considered efficient. In our DEX, we employ a system known as Liquidity Tracking Bits to maintain records of bins with available liquidity, enabling us to locate suitable bins for order execution efficiently.



The Liquidity Tracking Bits (LTB) represent a form of bitmap used to signify the presence or absence of liquidity within a given bin. In this scheme, if the bin situated at step n contains liquidity, the nth bit within the LTB is set to 1; conversely, if no liquidity is present, the corresponding bit is set to 0. As mentioned in **2.4**, with a maximum of 2^{21} bins, the requisite data size for LTB amounts to 2^{21} bits.

4.6 Window

Considering the technical limitations of TVM cells, which can store up to 1023 bits and 4 reference cells, it becomes evident that storing the Liquidity Tracking Bits (LTB) in a cell is unfeasible. Therefore, it becomes imperative to partition the bits into cell-sized segments appropriately. To address this, we introduce the concept of a "window" as a unit of this size. When the window is set to 1023 bits, the maximum number of keys required when using TVM's dictionary is 2^{11}. The appropriate window will be determined at a later stage. In cases where windows exhibit zero liquidity, there is no necessity to store any keys. In typical scenarios, the preponderance of liquidity is concentrated centrally, resulting in the absence of values for the vast majority of windows. Consequently, only a small number of keys are utilized.