HipoSwap - a high potential liquidity maker

L. Han, G. S, J. W, X. Z, C. L, S. L, T. Wei

research@hipo.com

August 27, 2020

Abstract

HipoSwap is a high potential AMM (Automated Market Maker) protocol for token exchange implemented using CPMM (Constant Product Market Maker). Besides the liquidity pool provided by CPMM, two additional maker pools are introduced in each X-Y trading pair, i.e, the X maker pool and the Y maker pool. These two maker pools not only provide a new channel of single-token exposure for both buyers and sellers to act as market makers, but also increase the market liquidity and significantly reduce price slippage for market takers.

Background

The blockchain industry has been developing rapidly in recent years, with thousands of blockchain assets/ tokens being issued every year. The growth trend keeps rising and far from reaching its peak. In the predictable future, millions of blockchain assets will co-exist, as every company or individual could issue their own assets. These blockchain assets, by its nature, are stored on their respective blockchain in a decentralized manner and users have strong demand for token swap. Since the trading volumes of blockchain assets are dominated by centralized cryptocurrency exchanges, users with demand of token swap have to transfer their tokens from personal wallet on decentralized blockchain to custodial wallet managed by centralized cryptocurrency exchange, and rely on the exchange's matching engine to cross their orders through either pending order bidding or call auction mechanism.
The following two concerns are raised for centralized cryptocurrency exchanges:

1) Users effectively lose their control over their blockchain assets once the assets are transferred to the custodial wallet managed by centralized cryptocurrency exchanges, and the safety of their assets will be possibly threatened whenever an issue occurs in the exchange.

2) Centralized cryptocurrency exchanges are limited in resources, and therefore, only capable of handling a limited number of assets and trading pairs. For example, Gate.io, a world leading cryptocurrency exchange, only supports around 300 assets and 500 trading pairs, which is far behind the amount of potential millions of assets in the future.

Decentralized exchanges can easily solve the two problems above, however, their solution is not flawless due to the following defects:

1) As the decentralized blockchain needs to verify and store every single transaction on the blockchain node, both of the processing and the storage logic must be simple enough to minimize the transaction cost. Therefore, the logic of order placing and order matching that used by traditional exchanges do not function well on decentralized exchanges. Some decentralized cryptocurrency exchanges have adopted an approach of off-chain orderbook and matching engine to facilitate order placing and order matching, which reduces the trading cost as only the trade settlements that need to be processed on-chain; however, this solution introduces other issues including manipulation risks and trust crisis due to poor decentralization.

2) It also poses difficulties in respect of blockchain asset safety when dealing with decentralized cryptocurrency exchanges. On a decentralized cryptocurrency exchange, individual users need to keep and store their private keys by themselves. Due to lacking of experience and infrastructure, individual users are more vulnerable to hacking attacks and prone to falling victims of private key theft. Furthermore, private keys that are stored by individual users are more likely to get lost or destroyed. To address this issue, multi-signature hardware device and vault account are introduced by GateChain, to reduce the hurdles for safe custody of decentralized assets, so that individual users are able to safely store their blockchain assets in their own hands.

Prior work

In 2017, Alan Lu proposed a decentralized asset exchange protocol suitable to be implemented on Ethereum [1], which deployed a constant product formula X*Y=k in a smart contract on Ethereum, to achieve automated market marker for the asset exchange of X and Y. Implemented by Uniswap, this mechanism has made a success in DeFi upsurge in 2020 [2]. The constant product formula can be further generalized to multiple assets with arbitrary weights. Balancer [3] protocol introduced a generalized constant product value function $V = \prod_t B_t^{W_t}$ to automatically configure a portfolio to maintain the initial proportionality in value

and provide liquidity for exchanging any two assets in the portfolio.

The decentralized automated market marker implemented using CPMM has its own defects as elaborated below:

1) Impermanent loss. Whenever the price deviates, either upward or downward, it incurs impermanent losses for liquidity providers, and such losses would not be recovered until the price returns to the origin of the price when the liquidity provider deposited. This inevitable impermanent loss greatly reduces the enthusiasm of liquidity providers in providing liquidity to the market. To address this issue, a higher transaction fee of 0.3% is charged to the market takers by Uniswap to reward the liquidity providers aiming to make up their

impermanent losses. However, higher transaction fee in turn affects the motivation and enthusiasm of the market takers.

2) High price slippage cost. When using a curve model like the constant product formula of X*Y=k, a high price slippage may occur for large volume transactions in particular, which would significantly increase the trading cost for market takers. A straight line model, such as X+Y=k, would eliminate the price slippage, however, the price in this model does not reflect the changes of demand and supply on the market. Curve.fi [4] proposed a new model that combines both the curve model and the straight line model to effectively reduce the price slippage in stable asset swap, but this new model is not suitable for the swapping of non-stable assets.

In 2020, Bancor proposed a new method in its V2 version of decentralized asset swap protocol, which uses an oracle machine to get prices from centralized exchanges and dynamically update the weight of its own liquidity pool, in order to eliminate the impermanent losses[5, 6] for liquidity providers. The V2 version also reduces price slippage by amplifying its own liquidity pool by 20 times. However, this method only fits into mainstream assets that have been listed on centralized exchanges and supported by an oracle machine. For tokens with small market capital, especially new tokens, it is difficult to get fast and comprehensive support from oracle machine. Therefore, the scope of application of this method is greatly limited.

Our Proposal: HipoSwap

We propose a new decentralized token swap protocol, the HipoSwap protocol, which is an AMM protocol for token swap that provides both lower price slippage and better market liquidity. Besides the liquidity pool provided by CPMM, two additional single-currency maker pools are introduced by HipoSwap in each X-Y trading pair, i.e, the X Maker Pool and the Y Maker Pool. These two maker pools effectively increase market liquidity and significantly reduce price slippage for market takers.

HipoSwap protocol consists of a CPMM similar to Uniswap protocol and Balancer protocol, which introduces a liquidity pool consisting of asset X and asset Y allocated according to pre-defined asset ratio, where the liquidity is provided by liquidity providers (LP); and another two new mutually opposed maker pools, i.e, the X Maker Pool and the Y Maker Pool, where the liquidity is provided independently by X Maker and Y Maker respectively.

The HipoSwap protocol employs a leverage setting to satisfy a minor portion (such as 10%) of market takers' needs using the liquidity pool provided by CPMM; where the major portion (such as 90%) would be satisfied by either X Maker Pool or Y Maker Pool by default. The Liquidity Pool will only be used in case of inadequate balances in the X or Y Maker Pools.

When a market taker initiates token swap from the liquidity pool (for example, 10% from the market takers' needs), the price will fluctuate due to changes in demand, and a reasonable price will be eventually formed via competition of Takers' operations from both buy-side and sell-side.

When a market taker initiates token swap from X or Y Maker Pools (for example, 90% portion from the market taker' needs), the market price will be determined by the current price of the Liquidity Pool to avoid larger price slippage. When a market taker swaps asset Y for asset X, the amount of asset X will be deducted from the X Maker Pool (denoted by Xm) and the amount of asset Y will be increased (denoted by Ymr); Meanwhile, the transaction fee rewards in the X Maker Pool, denoted by Yfee, will also get accumulated. Similarly, when a market taker swaps asset X for asset Y, the amount of asset Y will be deducted from the Y Maker Pool (denoted by Ym) and the amount of asset X will be increased (denoted by Xmr); Meanwhile, the transaction fee rewards in the Y Maker Pool, denoted by Xfee, will be accumulated.

After X market maker deposit asset X into the X Maker Pool, they are able to withdraw asset Y that to be exchanged from Ymr pool at anytime as long as Ymr has sufficient balance. The swap price will be calculated based on the average of historically accumulated swap price of asset Y in Ymr (price_xy_avg). In order to encourage market makers for asset X to deposit more X and stay longer, a time based handling fee system is designed in such a way that, if the deposit duration is less than the average deposit time of all market makers in the pool, a withdrawal fee will be charged and added to Yfee; if the deposit duration is greater than the average deposit time of all market makers in the pool, a rebate, taking from the Yfee pool, will be rewarded.

After Y market maker deposit asset Y into the Y Maker Pool, the market makers of asset Y can withdraw asset X that to be exchanged from Xmr pool at anytime as long as Xmr has sufficient balance. The swap price will be calculated based on the average of historically accumulated swap price of the asset X in Xmr (price_yx_avg). A fee/reward, taking from or adding to the Xfee pool, will be either charged or given subject to the deposit duration in the same method as described above.

A new Maker Pool needs to be created from time to time when certain conditions are met. The reason is that asset in a particular Maker Pool will be gradually swapped for the other assets. As time goes by, it is likely to have an average swap price in the Maker Pool less favorable comparing to the price from external markets. For example, the average BTC swap price at a Maker Pool is 11000 USD, while other markets have seen prices of 12000 USD or even higher. In this case, no user would like to continue depositing BTC to the Maker Pool for swap and as a result, the remaining BTC in the Maker Pool will be drained off soon. Such situation will not be eased until the BTC swap price in the Maker Pool matches the price from external markets. Otherwise, the exhausted Maker Pool will no longer be functioning, and might bring dramatic uncertainties to the liquidity of the Maker Pool. To address this issue, we propose a solution to switch the Maker Pool when certain conditions are met, such as the amount of asset remained in a particular Maker Pool is less than the pre-defined threshold. We also set a minimum duration for Maker Pool, to avoid frequent switching or accidental switching under some special circumstances. For example, a Maker Pool will only be allowed to trigger a switching after 30 minutes or the corresponding average number of blocks.

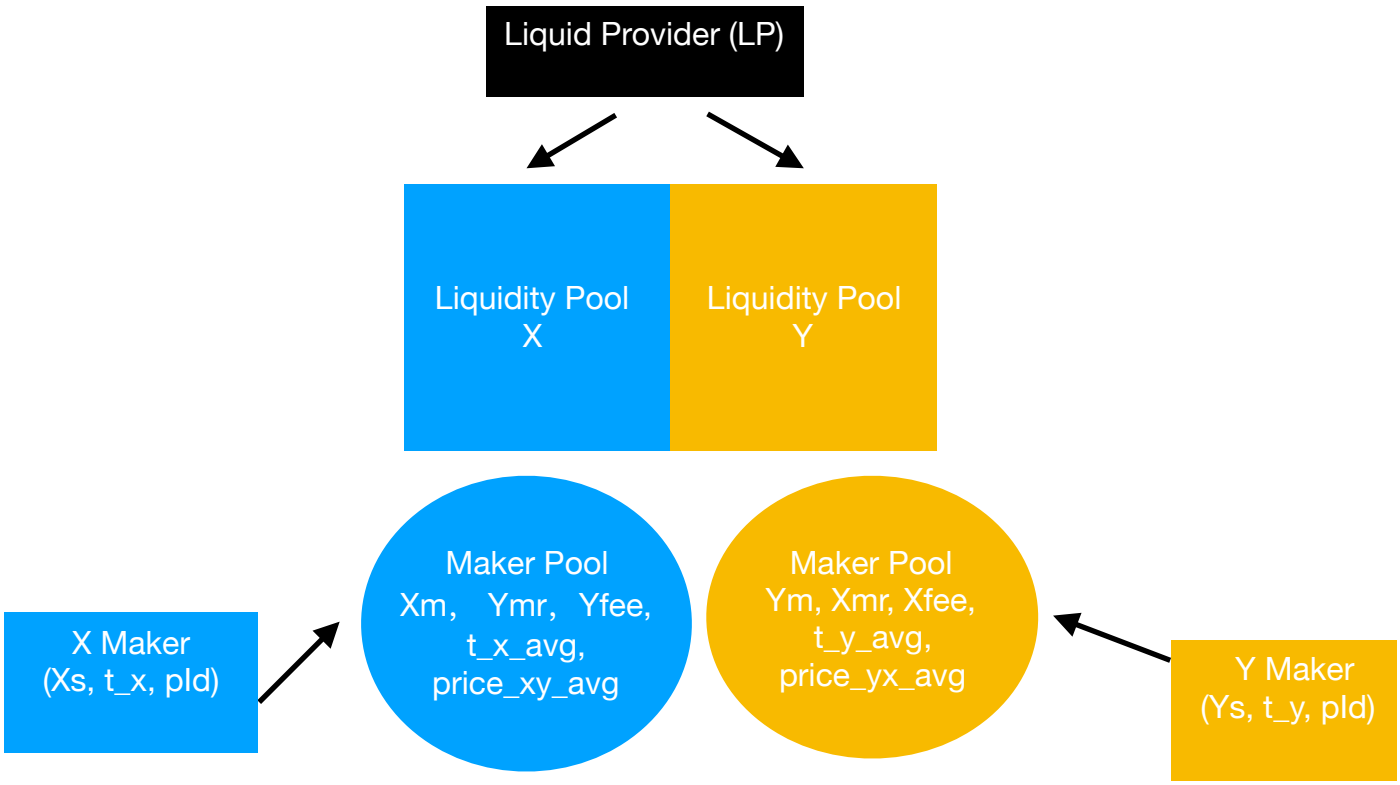
When the majority (such as 90%) of assets have been swapped from a particular Maker Pool, , the Maker Pool switching will be triggered and a new Maker Pool will be created to receive subsequent deposits from market makers. For example, when the ratio of remaining X value to swapped Ymr value in a X Maker Pool is less than 1:9, a new X Maker Pool will be created. All subsequent deposits of asset X will go into the new X Maker Pool, and the average swap price in the new Maker Pool will be re-calculated based on the new trades only. Makers who have deposited asset X to the previous X Maker Pool can only swap and withdraw assets from the previous Maker Pool. This solution is designed to settle the token swap for the former market makers from time to time without affecting the enthusiasm of the subsequent market makers in providing liquidity due to price fluctuations.

To reduce read and write operations from the storage in a smart contract, a random number can be generated to determine whether a newly placed order from market takers shall be swapped from the Liquidity Pool or the Maker Pools. This solution reduces the transaction cost for market takers, however, it might bring uncertainties to the price slippage during token swap.

Advantages of HipoSwap protocol

1) Effectively reduce price slippage during token swap: if the leverage parameter is set to n, it can reduce the price slippage to 1/n of that in a CPMM model.

2) Provide a new channel for both buyers and sellers to act as market makers to save the transaction cost by providing liquidity to the market, rather than letting all users to be takers to trade with the liquidity providers with higher transaction cost and larger price slippage.



Four liquidity pools are introduced in HipoSwap protocol:

Liquidity Pool (X-Y): reserves the staking assets from LP while maintaining their proportional relation in market value.

X Maker Pool: records the amount of asset X deposited by X Maker in Xm, the swapped asset Y in Ymr, and the collected maker fees in Yfee.

Y Maker Pool: records the amount of asset Y deposited by Y Maker in Ym, the swapped asset X in Xmr, and the collected maker fees in Xfee

Definition of the four types of market participants:

Liquidity Provider (LP): users who lock their holdings to yield earnings. They deposit two or more assets (of equal value or at a certain ratio) to the Liquidity Pool to support the liquidity and price discovery of CPMM. They can obtain part of the taker fees as earnings but would take impermanent loss due to price fluctuations.

X Market Maker: users who want to exchange asset X to asset Y. These users would deposit asset X into a X Maker Pool, and wait for their deposited asset X to be swapped automatically for asset Y over a period of time using market price. This process would take an estimated time up to 12 hours for active trading pairs accordingly to analysis on Uniswap. The market price is calculated based on the CPMM price at the time the taker operates.

When asset X being deposited into the X Maker Pool, two values of (Xs, t_x) will be stored in the contract, where Xs refers to the amount of asset X being deposited; and t_x refers to the last deposit time.

Y Market Maker: users who want to exchange asset Y to asset X. They deposit asset Y to Y Maker Pool and wait for their deposited asset Y to be swapped automatically for asset X over a period of time using market price. The whole process do take some time to complete.

When asset Y being deposited into the Y Maker Pool, two values of (Ys, t_y) will be stored in the contract, where Ys refers to the amount of Y being deposited, and t_y refers to the last deposit time.

Market Taker: Market taker initiates a swap request actively and get fulfilled instantly after the transaction is confirmed on the blockchain.

Below are definitions of some operations (here we use the x*y=k model from Uniswap as an example, a generalized constant product model can also be used as the one used in Balancer):

Liquid Provider Add Liquidity: addLiquidity(Δx) ,where asset X and asset Y are added to Liquidity Pool at the same rate:

$$X' = X * (1 + \alpha)$$
$$Y' = Y * (1 + \alpha)$$
$$\alpha = \frac{\Delta x}{X}$$

Liquid Provider Remove Liquidity: removeLiquidity(Δx), where asset X and asset Y are removed from Liquidity Pool at the same rate:

$$X' = X * (1 - \alpha)$$
$$Y' = Y * (1 - \alpha)$$
$$\alpha = \frac{\Delta x}{X}$$

X Maker Deposit : XMakerDeposit(Δx), increases Xm at X Maker Pool, and meanwhile, modifies debt and latest deposit time(Xs, t_x) of the market maker. If any asset X was deposited before and has not been withdrawn, the deposit time will be overwritten with the current time and the reward of withdrawal fee will be affected.

Please note, as there may be several X Maker Pools co-existing at the same time, the latest Pool Id will be retrieved and used to settle new deposits. If a user's current Pool Id is not the latest, a mandatory liquidation according to the default ratio will be required against the old pool before depositing to the new pool is allowed.

$$p_{Id} = Current\ X\ Pool\ Id$$
$$Xm'(p_{Id}) = Xm(p_{Id}) + \Delta x$$
$$Xs' = Xs + \Delta x$$
$$t_x' = current\ time$$

Create New X Maker Pool: Before depositing Δx to X Maker Pool, a check will be performed based on pre-defined conditions to determine whether a new Maker Pool needs to be created. When the ratio of Xm value to Ymr value is less than 1:9, a new X Maker Pool will be created. A minimum duration will be configured on the new Maker Pool in order to avoid frequent switching or accidental switching due to some special circumstances. For example, a Maker Pool must be used for at least 30 minutes (or the corresponding average number of blocks) before triggering the switching.

$$if(Xm's\ value < 1/9\ Ymr's\ value) : Current\ X\ Pool\ Id\ +=1$$

X Maker Withdraw: XMakerWithDraw(Δx, Δy), a market maker can decide whether to withdraw only part of asset X or part of asset Y, or to withdraw both asset X and asset Y in the same transaction as long as there are enough balances in Xm and Ymr. The withdrawal amount will be deducted from Xm, Ymr as well as the debt record (Xs), where Xs cannot be less than 0. Since token swap is not required when asset X is withdrawn from Xm, neither transaction cost nor rewards are provided. However, a time based transaction cost will be charged when asset Y is withdrawn from Ymr, and the price will be calculated based on the historically accumulated average swap price. If the deposit duration is less than the average duration of all market makers in the pool, a fee is charged and will be added to Yfee. If the deposit duration is longer than the average duration of all market makers in the pool, a rebate will be rewarded from Yfee.

Please note, the assets must be withdrawn from the Maker Pool, with the Pool Id matching with the record in the market maker's account; However, this Pool Id may not always be the latest.

Calculate the total debt to be deducted from the market maker:

$$\Delta x' = price\_xy\_avg * \Delta y + \Delta x$$

Here is the procedure of deduction calculation:

$$\begin{aligned} Xm'(p_{Id}) &= Xm(p_{Id}) - \Delta x' \\ Xs' = Xs - \Delta x';&only\ if\ \Delta x' <= Xs \\ Ymr'(p_{Id}) &= Ymr(p_{Id}) - \Delta y \\ Yr' = Yr - \Delta y;&only\ if\ \Delta y <= Yr \end{aligned}$$

For the case where Δx, Δy is not given as input parameters in the withdraw operation, both Δx and Δy will be calculated by the current proportion of Xm value and Ymr value in the X Maker Pool as:

$$\sigma = \frac{Xm}{(Xm + price\_xy\_avg * Ymr)}$$

Therefore, the total debt to be deducted from the market maker can be calculated as such:

$$\begin{aligned} \Delta x &= \sigma Xs \\ \Delta y &= (1-\sigma)Xs * price\_xy\_avg \\ \Delta x' &= Xs \end{aligned}$$

Here is the procedure of deduction calculation same as the general case:

$$Xm'(p_{Id}) = Xm(p_{Id}) - \Delta x'$$

$$Xs' = Xs - \Delta x'; only\ if\ \Delta x' <= Xs$$
$$Ymr'(p_{Id}) = Ymr(p_{Id}) - \Delta y$$
$$Yr' = Yr - \Delta y; only\ if\ \Delta y <= Yr$$

Y **Maker Deposit :** YMakerDeposit(Δy), increases Ym at Y Maker Pool, and meanwhile, modifies debt and latest deposit time (Ys, t_y) of the market maker. If any asset Y was deposited before and has not been withdrawn, the deposit time will be overwritten with the current time and the reward of withdraw fee will be affected.

Please note that as there may be several Y Maker Pools co-existing at the same time, the latest Pool Id will be retrieved and used to settle new deposits. If a user's current Pool Id is not the latest, a mandatory liquidation according to the default ratio will be required against the old pool before depositing to the new pool is allowed.

$$p_{Id} = Current\ Y\ Pool\ Id$$
$$Ym'(p_{Id}) = Ym(p_{Id}) + \Delta y$$
$$Ys' = Ys + \Delta y$$
$$t_y' = current\ time$$

Create New Y Maker Pool: Before depositing Δy to Y Maker Pool, a check will be performed based on pre-defined conditions to determine whether a new Y Maker Pool needs to be created. When the ratio of Ym value to Xmr value is less than 1:9, a new Y Maker Pool will be created. A minimum duration will be configured on the new Maker Pool in order to avoid frequent switching or accidental switching due to some special circumstances. For example, a Maker Pool must be used for at least 30 minutes (or the corresponding average number of blocks) before triggering the switching.

$$if(Ym's\ value < 1/9\ Xmr's\ value) : Current\ Y\ Pool\ Id\ += 1$$

Y Maker Withdraw: YMakerWithdraw(Δy, Δx), a market maker can decide whether to withdraw only part of asset X or part of asset Y, or to withdraw both asset X and asset Y in the same transaction as long as there are enough balances in Ym and Xmr. The withdrawal amount will be deducted from Ym, Xmr as well as the debt record (Ys), where Ys cannot be less than 0. Since token swap is not required when asset Y is withdrawn from Ym, neither transaction cost nor rewards are provided. However, a time based transaction cost will be charged when asset X is withdrawn from Xmr, and the price will be calculated based on the historically accumulated average swap price. If the deposit duration is less than the average duration of all market makers in the pool, a fee is charged and will be added to Xfee. If the deposit duration is longer than the average duration of all market makers in the pool, a rebate will be rewarded from Xfee.

Please note, the assets must be withdrawn from the Maker Pool, with the Pool Id matching with the record in the market maker's account; However, this Pool Id may not always be the latest.

Calculate the total debt to be deducted from the market maker:

$$\Delta y' = price\_yx\_avg * \Delta x + \Delta y$$

Here is the procedure of deduction calculation:

$$\begin{aligned} Ym'(p_{Id}) &= Ym(p_{Id}) - \Delta y' \\ Ys' &= Ys - \Delta y'; only\ if\ \Delta y' <= Ys \\ Xmr'(p_{Id}) &= Xmr(p_{Id}) - \Delta x \\ Xr' &= Xr - \Delta x; only\ if\ \Delta x <= Xr \end{aligned}$$

For the case where Δx, Δy is not given as input parameters in the withdraw operation, both Δx and Δy will be calculated by the current proportion of Ym value and Xmr value in the Y Maker Pool as:

$$\sigma = \frac{Ym}{(Ym + price\_yx\_avg * Xmr)}$$

Therefore the total debt to be deducted from the market maker can be calculated as such:

$$\begin{aligned} \Delta y &= \sigma Ys \\ \Delta x &= (1-\sigma) Ys * price\_yx\_avg \\ \Delta y' &= Ys \end{aligned}$$

Here is the procedure of deduction calculation same as the general case:

$$\begin{aligned} Ym'(p_{Id}) &= Ym(p_{Id}) - \Delta y' \\ Ys' &= Ys - \Delta y'; only\ if\ \Delta y' <= Ys \\ Xmr'(p_{Id}) &= Xmr(p_{Id}) - \Delta x \\ Xr' &= Xr - \Delta x; only\ if\ \Delta x <= Xr \end{aligned}$$

Processing swap request from market takers

Let's define the leverage factor in this proposal to be n. Theoretically, the proportion of quantity to swap from the Liquidity Pool and the Maker Pool should be $r_s = \frac{1}{n}$ and $r_m = 1 - \frac{1}{n}$respectively, where $\gamma = 1 - \rho$ and ρ is the taker fee.

In the case where market taker swapping Δx for Δy

The latest swap price in Liquidity Pool is p=x/y

At first, obtain Y from Y Maker Pool, such as (transaction fee included):

$$\Delta y_1 = Min(\Delta x / p * r_m,\ Ym) * \gamma$$

The amount of swapped X is:

$$\Delta x_1 = p * \Delta y_1 / \gamma$$

The remaining amount of X to be swapped is:

$$\Delta x_2 = \Delta x - \Delta x_1$$

The remaining amount of X should be swapped to Y from the Liquidity Pool (same behaviour as Uniswap):

$$\Delta y_2 = \frac{\alpha\gamma}{1+\alpha\gamma} y$$

$$\alpha = \frac{\Delta x_2}{X}$$

Update the amount of Y remained in Y Maker Pool, as the following:

$$Ym' = Ym - \Delta y_1 / \gamma$$

$$Xmr' = Xmr + \Delta x_1$$

$$price\_yx\_avg' = \frac{price\_yx\_avg * Xmr + \Delta y_1}{Xmr + \Delta x_1}$$

Update the amount remained in the Liquidity Pool, as the following:

$$Y' = Y - \Delta y_2 / \gamma$$

$$X' = X + \Delta x_2$$

In the case where market takers swapping Δy for Δx

The latest swap price in Liquidity Pool is p=x/y

At first, obtain X from X Maker Pool, such as (transaction fee included):

$$\Delta x_1 = Min(\Delta y * p * r_m,\ Xm) * \gamma$$

The amount of swapped Y is:

$$\Delta y_1 = \Delta x_1 / p / \gamma$$

The remaining amount of Y to be swapped is:

$$\Delta y_2 = \Delta y - \Delta y_1$$

The remaining amount of Y should be swapped to X from the Liquidity Pool (same behaviour as Uniswap):

$$\Delta x_2 = \frac{\beta}{1-\beta}\frac{1}{\gamma}x$$

$$\beta = \frac{\Delta y_2}{Y}$$

Update the amount of X remained in X Maker Pool as the following:

$$Xm' = Xm - \Delta x_1 / \gamma$$

$$Ymr' = Ymr + \Delta y_1$$

$$price\_xy\_avg' = \frac{price\_xy\_avg * Ymr + \Delta x_1}{Ymr + \Delta y_1}$$

Update the amount remained in the Liquidity Pool as the following:

$$X' = X - \Delta x_2 / \gamma$$

$$Y' = Y + \Delta y_2$$

An optional interface to trigger the creation of new X Maker Pool

Normally the new Maker Pool creation will be triggered at certain conditions when a deposit is initiated from the market makers. In case the creation of new Maker Pool is failed to be triggered promptly when the condition is met, anyone can call this interface to trigger it manually.

When the creation request of new X Maker Pool is received, the system must verify whether the ratio of Xm value to Ymr value is less than 1:9 before processing it; otherwise, no action will be taken.

$$if(Xm's\ value < 1/9\ Ymr's\ value) : Current\ X\ Pool\ Id\ +=1$$

An optional interface to trigger the creation of new Y Maker Pool

Normally the new Maker Pool creation will be triggered at certain conditions when a deposit is initiated from the market makers. In case the creation of new Maker Pool is failed to be triggered promptly when the condition is met, anyone can call this interface to trigger it manually.

When the creation request of new Y Maker Pool is received, the system must verify whether the ratio of Ym value to Xmr value is less than 1:9 before processing it; otherwise, no action will be taken.

$$if(Ym's\ value < 1/9\ Xmr's\ value) : Current\ Y\ Pool\ Id\ +=1$$

Conclusion

The HipoSwap protocol integrates a CPMM similar to the one implemented in Uniswap protocol, which can automatically discover market prices without relying on the price feeds from oracle machine. However, the CPMM used in Uniswap protocol does not have an interface for market makers, except for the liquidity provider; therefore, all the other buyers and sellers using Uniswap protocol are forced to be market takers, which will result in higher price slippage and more trading cost especially for large-volume traders. The HipoSwap protocol offers a new channel for the buyers and sellers to act as market makers, so that better market liquidity could be provided by those large-volume traders who are not in a demand for immediate executions. In this way, market makers will not only effectively reduce their price slippage, but also earn trading fee rebate from market takers via providing market liquidity. Furthermore, this protocol also significantly reduces the trading cost for market makers, when price fluctuation is not taking into consideration. Meanwhile, due to the existence of market makers, who enable the possibility for market takers to get direct liquidity from both the CPMM Liquidity Pool and the Maker Pools, the price slippage will be effectively reduced. It has been calculated and proved that with the leverage of 10x, the taker price slippage could be reduced to 1/10 comparing with that only using the CPMM Liquidity Pool, given sufficient liquidity in the Maker Pool. In Uniswap protocol, a 0.3% fee is charged from market takers in order to compensate for the impermanent loss from liquidity providers due to price fluctuation. In the HipoSwap protocol, the overall trading fee could be effectively reduced for market takers with the implementation of an optimized fee structure. For example, a 0.35% trading fee could be charged for the amount directly swapped from the Liquidity Pool and used as a reward for liquidity providers to increase their enthusiasm; and a 0.15% trading fee could be charged for amount directly swapped from the Maker Pool, to reduce the cost for market takers. As majority of the transactions from market takers would be handled directly by the Maker Pool, such proposed fee structure would significantly reduce the overall trading cost for market takers. Ideally, assuming 90% of the amount is swapped from the Maker Pool, the average trading fee could be reduced to as low as 0.17%.

The operating mechanism of the Liquidity Pool implemented in the HipoSwap protocol is consistent with the Uniswap protocol and the Balancer protocol. When processing operations from the market takers, the smart contract on Uniswap or Balancer protocol could be invoked to facilitate the token swap and to retrieve the latest prices; operations from the Maker Pool will only be processed subsequently. This mechanism is designed to fully utilize the market liquidity provided by the Liquidity Pool in Uniswap.

In order to incentivize market makers and liquidity providers to provide more liquidity, a governance token could be issued and rewarded to the liquidity providers as liquidity mining rewards. The governance token, that mined via providing market liquidity, can be used to participate in voting for proposals of future system upgrade; and as reserve for system fee rewards and etc.

The HipoSwap protocol can be implemented and deployed on Ethereum, Tron and EOS blockchain using Smart Contracts; it can also be implemented on GateChain using native codes. The implementation on GateChain could combine the features of both the Regular account and the Vault account provided by GateChain that store commonly used small amount of funds in the Regular account and infrequently used large amount of funds in the Vault account, to balance the transaction demand and asset security requirements of users.

If the HipoSwap protocol is to be deployed on a blockchain with low network throughput and high network transaction costs, such as Ethereum, the Layer 2 protocols could be implemented together to increase the efficiency of token transfer and reduce the transaction costs.

References：

[1] 2017, Alan Lu, Building a Decentralized Exchange in Ethereum https://blog.gnosis.pm/building-a-decentralized-exchange-in-ethereum-eea4e7452d6e

[2] 2020, Uniswap v2 Core: https://uniswap.org/whitepaper.pdf

[3] 2019, A non-custodial portfolio manager, liquidity provider, and price sensor. https://balancer.finance/whitepaper/

[4] 2019, StableSwap - efficient mechanism for Stablecoin liquidity, https://www.curve.fi/stableswap-paper.pdf

[5] 2020, Bancor V2 Launches: https://blog.bancor.network/bancor-v2-launches-b1fec492eeb2

[6] 2020, Breaking Down Bancor V2 Dynamic Automated Market Makers https://blog.bancor.network/breaking-down-bancor-v2-dynamic-automated-market-makers-4e90c0f9a04