Cauldron protocol: stability via volatility

Benjamin Wang

25th May 2022

Abstract

The following describes a protocol that aims to create a stable asset (sToken) through shares of a basket of volatile assets i.e. stability via volality. We do this by creating a two sided market: on one side, options traders seeking volality, and on the other, stablecoin holders seeking stability. This yields both, a highly capital efficient options market and a stablecoin that, over time, achieves native stability without any reference to any fiat currency, index or external price data.

Each option is created by depositing collateral into a pool and receiving back an amount of sToken equal in (economic) value to the amount of the asset they deposited (100% collateralisation). The value of the sToken changes depending on the value of all the collateral deposited. At maturity, the depositor can decide whether to keep their sTokens or repay the sTokens to regain their collateral (no liquidations based on external market movements). After maturity, abondoned collateral is auctioned for sTokens.

The base mechanism of the pool is a Balancer-type pool where only single-sided LP deposits and withdrawals are allowed i.e. the sToken is effectively the LP share for the pool. Price oracles are not needed since the offered price is always being arbitraged back to the market price. Meanwhile the asset weights of the pool slowly change over time based on the total maturity allocated for each asset type.

The result is a pool of non-correlating, competing assets. As more and more assets and participants participate, the pool becomes more and more stable. The sToken represents a share of this pool.

Initially, the sToken will be unstable and the incentive to hold will be a high ROI crypto asset with a very low risk profile. Overtime, the sToken will become more stable.

An asset list determines which ERC20 addresses can be used in the protocol. Decentralised governance is used to add new assets and also remove assets in the case that they have been corrupted.

Unlike current offerings, this has the potential of creating a much more decentralised stablecoin without any external dependencies such as price oracles or index data.

1 Motivation

Current stablecoin offerings have at least one of the following downsides:

- External oracle solution with unclear security models.
- Stability in reference to fiat currency.
  - Necessitates the use of centralised exchange data or centralised, custodied stablecoins.
  - Forces holders to expose themselves to inflating fiat currency.
- Complex and/or insecure governance processes.
- Over-collateralisation creating capital inefficiency and no possibility for closed-cycle arbitrage [2].
- Collateral liquidation based on external market value putting additional risk on the depositor.
- Use a basket of arbirarily chosen assets to create a stability but with no crypto-economic mechanism to choose stabalising assets.

The current offering aims to solve all of these issues.

2 Stability via volatility

Looking at the methods of referrencing stability in Defi one might conclude that fiat currencies or real world indexes are the only measures of stability. But, in fact stability is a relative measure. Put simply: a portfolio of assets can be made more stable by adding additional assets which do not correlate with the others. As you add more non-correlating assets, greater stability can be achieved. This is similar to the way fiat currencies achieve stability - redeemability for all assets and debts in the economy.

This is exactly how the Cauldron protocol aims to achieve stability - an increasingly larger basket of non-correlating assets.

2.1 Incentivising non-correlating assets

But how can we incentivise participants to add non-correlating assets and remove correlating ones? The answer is that depositors are actually effectively borrowers/options-buyers and are therefore incentivised to deposit such assets. When they deposit an asset in exchange for a share of the basket, they are in fact borrowing the share of the basket and depositing their asset as collateral. If they are always able to redeem the same amount of shares for the same amount of collateral, they retain exposure to it. Therefore, depositors can use the protocol as a way of going long. And, given that it is perfectly collateralised (rather

than over-collateralised), they can do this multiple times for the same asset, therefore increasing leverage and exposure.

The conclusion from this is that depositors will naturally add assets which they expect to increase in value relative to value of the whole basket. They may also use it as a hedge in the case that the asset goes down in value, however, they would only ever pay fees to hedge an asset which they expect to increase in value.

It is clear that depositors will always seek to add non-correlating assets since any correlating asset would serve no purpose to deposit and pay a fee.

2.2 Defaults

Defaults (or abandoned collateral) will be very common since the collateralisation ratio is only 100% and therefore, depositors will not withdraw assets that have gone down in value relative to the pool. To purge itself of these assets and repay the debt, the remaining collateral is auctioned off but of course there will still be outstanding debt which is un-repaid. To recover this remaining debt, deposit fees are set equal to the average loss (within a range of maturities).

3 Protocol overview

- Asset pool module
  - Deposit
  - Withdraw
  - Auction bid
  - cToken to sToken swaps
- Governance module
  - Adding assets to the whitelist
  - Removing assets from the whitelist
  - Inflation funding proposals

4 Theory

Here we outline the general theory for how we may calculate the correct price between a share of a pool of assets (sToken) and an amount of specific collateral asset.

As first implemented by Uniswap and later generalised by Balancer [1], a constant value function may be defined by:

$$V = \prod_t B_t^{W_t} \tag{1}$$

where t refers to the asset type, B is the amount of each asset and W is the normalised weight for each asset, such that the sum of all weights equals 1.

It can be shown [1] that while V is invariant and as long as there are rational market actors, the spot price will track the market price. And furthermore, if that is true, the weight of each asset is equal to its share of value relative to the total pool value. So by changing the weights, we may redefine the relative amount of each asset in terms of value.

So for adding an amount of a specific asset we can calculate the amount of pool shares returned as [1]:

$$P_{issued} = P_{supply} \cdot ((1 + \frac{A_t}{B_t})^{W_t} - 1) \tag{2}$$

where P_{issued} is the amount of pool shares given out by the pool, P_{supply} is the total supply of pool shares, A_t is the amount asset type t added to the pool and B_t is the amount of asset type t already in the pool.

When the spot price is at the market price, equation 2 gives the depositor an equal value of pool shares in exchange for small deposits. As the deposit size increases the depositor gets a lower amount of value relative to the value they deposit. I.e. asset deposits affect the spot price similarly to dex trades such as on Balancer. This demonstrates that arbitrage can take place by single asset deposit only i.e. without direct asset-asset trades. Summary so far:

- Depositor receives shares of approximate equal value relative to the pool when depositing an asset.
- Relative economic value of the total amount of each asset in the pool can be set by the asset weights.
- Arbitrage is used to revert the pool back to market prices without using asset-asset trading.
- By preventing direct asset-asset trading, depositors can always redeem their pool shares for exactly the same amount of the asset they deposited.

However, we are now left with the problem of how and on what basis to change the weights of the pool. Since we want the pool value to be stable over some timescale, we can change the weight based on the maturity that depositors decide on. I.e the maturity indicates the time frame over which the depositor has some confidence that the collateral will out-perform the rest of the pool. Therefore, we can weight the pool by the length of the maturity for each asset type:

$$z_t = \sum m_i d_i \tag{3}$$

$$w_t = \frac{z_t}{\sum z_t} \tag{4}$$

where z is the absolute weight value, t refers to each asset type, i refers to each deposit, m refers to maturity, d refers to debt and w refers to the relative weight.

To avoid sudden changes to the weights or any manipulation a 24 hour moving average is used in the implementation of equation 2.

Finally, the maturity products incrementally decay over time for all assets so that the weights are based on the maturity product remaining for all deposits rather than the maturity product when they were created:

$$z_t^1 = z_t^0 - d_t(x^1 - x^0) \tag{5}$$

where x refers to time.

4.1 Maturity fees

As mentioned in section 2.2, deposit fees need to be charged to recoup the losses that occur from depositors defaulting on their collateral. Since the likely loss correlates with the maturity, greater maturities should charge greater fees. Therefore, maturities are divided up into tranches and the average loss ratio is recorded for each tranche. We can then interpolate between the nodes of the tranches to apply the correct fee for any maturity.

5 Protocol functions

5.1 Depositing

The depositor chooses the asset amount, asset type and maturity for the deposit. As described in section 4, the amount of debt created for a given amount of a specific asset can be calculated with equation 2 which can then be used to calculate the amount of fees as in section 4.1. Fees are deducted from the debt to calculte the amount of sTokens minted.

5.1.1 Debt-share-change limit

For safety, the rate of increase for the relative debt-share of each asset type is limited:

$$d_s = \frac{d_t}{\sum d_t} \tag{6}$$

$$d_{\text{changeLimit}} > \frac{d_s}{d_s^a} - 1 \tag{7}$$

where d_s is the relative debt-share, d_t is the total debt for asset type t and d_s^a is the moving average for d_s. This limits protocol losses should an asset type be compromised e.g. by a centralised party minting arbitrary amounts of it.

The debt-share-change limit does not limit the scalability of the protocol since it is crucially only the *share* of the debt which is limited rather than the absolute amount.

Finally, an NFT is minted for accessing the vault during withdrawal.

5.2 Withdrawing

5.2.1 Arranging withdrawal and priming

A withdrawal can be made at any time by the vault NFT holders. However, if collateral is removed instantly, this suddenly changes the collateral price set by equation 2. The offered price will ultimately be arbed back to the market price but this does result in a loss of value for the protocol. To minimise this, a priming period starting 24 hours before the withdrawal time where increasing amounts of collateral are removed from the pool (but not the vault) changing the offered price. The amount of the withdrawal that can be primed increases linearly with time over the 12 hour period. Therefore, price impact and arbitration losses are minimised.

Therefore, first the withdrawer sets the withdrawal amount and withdrawal expiry. Then 12 hours before the withdrawal expiry they can prime their withdrawal and then withdraw it. After withdrawal expiry, any remaining withdrawal amount is auctioned *despite it being before the deposit expiry.*

5.2.2 Auction bidding

Collateral is auctioned if the withdrawal expiry or the vault expiry have expired with remaining withdrawal or vault collateral, respectively. This is achieved with a decreasing dutch auction starting at the vault price.

5.2.3 After withdrawing or bidding

As decribed in section 2.2, the vault debt is then set to zero and also deducted from the total debt. Similarly, as described in section 4.1, maturity fee average is updated for the appropriate maturity tranche node.

5.3 cToken-sToken swap mechanism and goverance fees

This sub-section is not necessary for understanding the Asset Pool module and can be skipped if the reader wishes.

In order for the cToken governance token to secure the protocol, the cToken supply needs to be at least as valuable as the debt which it is securing. However, protocol governance tokens often wildly vary in market price regardless of the protocol fundamentals, especially in the early bootstrapping stages. As with many governance tokens in Defi, the cToken derives its value from the future cash flow from governance fees (charged in addition to maturity fees). Many Defi protocols set this fee arbitrarily, however, in this protocol a novel mechanism has been built which sets fees based on the moving averages of the cToken and sToken market caps.

We first define a swap mechanism between cTokens and sTokens in order to record the changes in price. We need no liquidity since both tokens can be burned and minted.

$$c^2 f + s^2 = k \tag{8}$$

where c and s are the respective token supplies, f is a swap factor and k is a constant. (Note: this is similar to the generalised AMM invariant (equation 1) except that we must switch the sign of the differentials because the trader is burning and minting supplies rather than adding and removing liquidity. This results in the equation for an ellipse.)

The price p_c of cToken in terms of sToken can be calculated as:

$$p_c = \frac{cf}{s} \tag{9}$$

Therefore, cToken market cap in terms of sTokens can be calculated as:

$$M_c = \frac{c^2 f}{s} \tag{10}$$

And therefore:

$$\frac{M_c}{M_s} = \frac{c^2 f}{s^2} \tag{11}$$

The swap factor, f, allows the token supplies to change during deposit and withdrawals while minimising arbitrage profits. To do this we make the assumption that there is a fair market value for the market capitalisation of the token supplies which, on average, remains constant before and after either deposit and withdrawals. Therefore, M_c should remain constant. From equation 10:

$$\frac{c_0^2 f_0}{s_0} = \frac{c_1^2 f_1}{s_1} \tag{12}$$

where 0 and 1 refer to before and after the sToken supply changes, respectively. Since cToken supply is not changing $f1$ can be calculted as:

$$f_1 = \frac{s_1 f_0}{s_0} \tag{13}$$

This also means that the swap price P_c remains constant.

When depositors pays governance fees the situation is slightly more complex:

- The depositor is minted the vault debt amount minus the fee. This effectively burns the fee amount of sToken.
- When the depositor needs to repay the debt amount of sToken, additional sToken will need to be minted by burning cToken.
- The depositor will need to buy this cToken from current holders therefore creating a a cashflow for cToken holders.

For burning cToken for sToken in the swap mechanism, after deposit but before withdrawal:

$$c_0^2 f_1 + s_0'^2 = c_1^2 f_1 + s_1^2 \tag{14}$$

where s_0' is the supply of sToken after the deposit but before the token swap. Solving equations 12 and 14 for f_1 yields:

$$f_1 = \frac{f_0 s_1}{s_0} + \frac{s_1^2 - s_0'^2}{c_0^2} \tag{15}$$

The gov fee for depositors is calculated using equation 11:

$$G_{fee} = d \cdot (\frac{c^2 f}{s^2} - 1) \tag{16}$$

where d is the sToken debt amount. In practice the moving average of the supply amounts are used to avoid fee manipulation. This means that the protocol attempts to make the market cap. of the corresponding token supplies equal and even results in a kind of liquidity mining (negative fees) where the market cap. of cToken is greater than that for sToken.

Using this kind of mechanism to calculate fees minimises the governance attack surface and better optimises fee values.

6 Governance

The goverance token has only two purposes:

- Add and remove assets from the whitelist.
- Provide future protocol-related funding.

Fundamentally, the only task necessary for cToken holders is to add or remove asset types from the whitelist where asset types correspond to ERC20 addresses. Inflation funding will help the protocol grow but it is not necessary for the security and functioning of the protocol.

This minimal simple set of voteable actions for governance makes it much easier to get participation rather than apathy as with many governance tokens today. Additionally, this minimises the attack surface.

Note: cToken holders do not need to express an opinion on the investment proposition of any asset, they only need express an opinion about whether an asset has been corrupted or not (see below).

6.1 Adding and removing assets

6.1.1 Asset corruption

The primary purpose of using a whitelist is to protect against adding assets which a centralised entity seeks to arbitrarily mint in order to extract value

from the Cauldron protocol. We refer to such assets as corrupted. Therefore, the primary responsibility of cToken holders is to prevent corrupted assets from being on the whitelist. We assume an un-corrupted asset can become corrupted therefore cToken holders must remove assets that become corrupted as well as adding un-corrupted ones.

In addition to this, all deposits are limited by in how fast the asset type can increase its relative share of the total debt compared with a 24 hour rolling average. This greatly restricts the amount of value an attacker can extract before cToken holders remove the corrupted asset.

Note that adding and removing assets types from the whitelist purely refers to halting new deposits for those asset types. It does not mean liquidating any collateral.

Adding and removing assets proceeds through coinvoting. However, a novel mechanism is used to both secure the protocol against vote bribing and also allow as few votes as possible to pass a proposal.

$$T_a = w_i c n_a \tag{17}$$

$$T_r = w_i c n_r \tag{18}$$

where T (in addition to quorum) is the minimum number of yes-votes to pass a proposal, w_i is the current weight or proposed initial weight, c is the cToken total supply, n is the safety factor to protect against potential attacks, a refers to proposals for adding new collateral types and r refers to proposals for removing collateral types. Since there is no profitable attack for removing assets, $n_r = 1$. In contrast, since the cost of borrowing or bribing might be very low, $n_a = 10,000$. This means that initial deposits will be very small, however, this is not a problem since the limit on increasing the debt doubles daily. In addition to this:

- Any attempt to pass a proposal must first allow 24 hours for the other side to attempt to over-turn the vote.
- A minimum quorum of 10% of staked cToken is required to pass any proposal.
- A super-majority of 67% of the votes is required to pass an add proposal or reject a remove proposal.

6.2 Funding proposals

A request for funding can be made in the form of a funding proposal. The funds are created by minting new cToken and inflating the cToken supply. In order to minimise any corruption, the amount of funds available are limited to a small percentage starting at 4.5% in the first round and going down to 0.2% per round after 3 years. By limiting the amount of funds to these amount, we assume that there will be legitimate proposals that can increase the value of the

cToken token by these amounts. Therefore, it is rational for attackers to fund these legitimate proposals rather than themselves.

6.3 Delegation

Given the level of voter apathy in coin-voting governance systems, delegated voting is a natural solution that also has some precedent in non-blockchain governance systems. Therefore, this should be encouraged. However, it should not be forced at the core protocol level, since there is no way to do this without requiring a minimum token amount for delegates which prevents new delegates from getting started.

Instead delegation can be implemented as external contracts to the protocol and has the added benefit of allowing delegates to be creative in the kinds of delegation models they and delgators want to use.

7 Initial growth and development

7.1 Organic liquidity mining

If cToken demand is strong, initial fees will be negative and the pool will grow very quickly in assets until the value of sToken market cap exceeds the cToken market cap which means that cToken will start being traded for sToken again and fees will become positive. This is essentially organic liquidity mining since cToken is always diluted to incentivise the creation of more debt until the total debt is equal to the value of the cToken supply.

7.2 Trader perspective

From the perspective for traders to use the protocol, they essentially have nothing to lose except for the small deposit fee. If they think the project will not succeed they can short the sToken by depositing their own assets in the asset pool. This creates an anti-fragile dynamic for the value of the sToken since it gains from negative sentiment.

7.3 Arbitrage price

The price of the sToken should tend to the arbitrage price which is the value of the corresponding share of the asset pool. E.g. if sToken is trading at above the value determined by the asset pool, arbitrageurs can deposit assets and, in exchange, receive a higher amount of sToken than on the secondary markets. Similary, if sToken is trading below its backed asset value, sToken can be bought from the secondary market and deposited into the asset pool to then get a greater amount of sToken.

It's worth noting that both of these strategies form a closed arbitrage cycle [2] which means they can be profitably repeated. This then means that if there's excess demand for the sToken, causing over-valuation on the secondary market,

the asset pool can scale up to meet demand. Similarly, bank-run crises are avoided since sTokens can always be redeemed for the collateral in the pool.

References

[1] Balancer Whitepaper (https://balancer.fi/whitepaper.pdf)

[2] Maker Dai: Stable, but not scalable (https://uncommoncore.co/maker-dai-stable-but-not-scalable)