Cauldron: An efficient Constant Product Market Maker Contract on Bitcoin Cash Through Micro-Pools

Dagur Valberg Johannsson dagurval@pvv.ntnu.no https://cauldron.quest

Abstract

This paper introduces Cauldron, a novel implementation of Constant Product Market Maker (CPMM) contracts on Bitcoin Cash using a micro-pools approach. By harnessing the inherent properties of Bitcoin Cash, particularly the UTXO model's ability to hold tokens and BCH simultaneously, Cauldron enables decentralised liquidity pools tailored for the Bitcoin Cash ecosystem. The micro-pools approach circumvents the need for a global state by allowing individual users to operate their own CPMM contracts, offering unparalleled scalability. The absence of a Decentralised Exchange (DEX) in Bitcoin Cash highlights the significance of Cauldron as an essential building block for fostering a vibrant token ecosystem. Through micro-pools, Cauldron efficiently facilitates decentralised trading, which is particularly critical for a blockchain network that is actively accommodating an expanding suite of tokens.

Introduction

In recent years, Constant Function Market Makers (CFMMs) have gained popularity within decentralised finance (DeFi) as an innovative solution for providing liquidity in digital asset exchanges. CFMMs utilise a deterministic pricing rule, altering the process of liquidity provisioning for market making and democratising access to global capital pools [1].

As of this writing, Bitcoin Cash, known for its low transaction fees and scalability, does not have a native Decentralised Exchange (DEX). This paper introduces Cauldron, a CPMM implementation on Bitcoin Cash that leverages the network's unique properties.

Cauldron employs a novel micro-pools approach, allowing individual users to operate small-scale CPMM contracts. This design, taking advantage of Bitcoin Cash's UTXO model and capability to hold tokens and BCH in the same UTXO, aims for scalability and efficiency, fostering the growth of the token ecosystem on Bitcoin Cash.

The paper further details the preliminaries of Bitcoin Cash essential for Cauldron, the contract structure, and potential applications.

Bitcoin Cash Preliminaries

Bitcoin Cash employs the Unspent Transaction Output (UTXO) model. A notable feature is the capability of UTXOs to store both tokens and BCH. In Cauldron, this is exploited to ensure that the UTXO contains quantities of tokens and BCH satisfying the CPMM formula. Moreover, Bitcoin Cash's scripting language supports introspection, a crucial building block for the contract as it combines native tokens with introspection capabilities to enforce the CPMM constraints.

CPMM Contract Structure

Micro-Pools Concept

This implementation deviates from the traditional single large liquidity pool model by utilising multiple small-scale CPMM contracts called micro-pools. Each liquidity provider operates their own set of micro-pools.

Constant Product Property

The contract adheres to the constant product formula, K = x * y, where K is the constant value, and x and y represent the quantities of tokens and BCH, respectively.

Contract Encoding and Logic

Coded in Bitcoin Cash Script, the contract enforces the constant product property. It employs introspection to ensure that the UTXO, when spent in a trade, is reconstituted in one of the transaction outputs with the constant product maintained, accounting for a small fee that accrues to the liquidity provider.

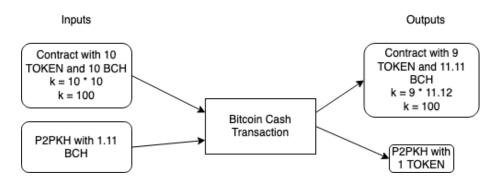


Figure 1: A trade where 1 token is purchased for BCH.

The example in figure shows a transaction trading 11.12 BCH for 1 TOKEN. First input of the transaction is a cauldron contract, and the first output replicates the

contract, rebalanced but maintaining K. The second input is the BCH funding the trade and the second output is the token traded.

For withdrawing the contract, the original creator of the contract is able to bypass the CPMM constraint by providing his public key and signature.

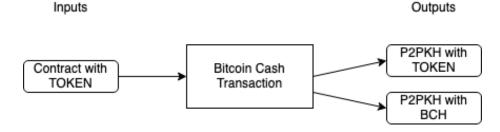


Figure 2: Contract owner withdrawing all funds.

A typical withdrawal would be taking the contract as input, providing public key and signature, then having one output each for token and BCH to the user. The full contract code is provided as an appendix.

Aggregating Micro-Pools

The Cauldron contract allows for aggregation of micro-pools to increase liquidity for trades. Multiple contracts can be used as inputs, combining their liquidity. This enables a trade to be executed as though it were interacting with a larger pool.

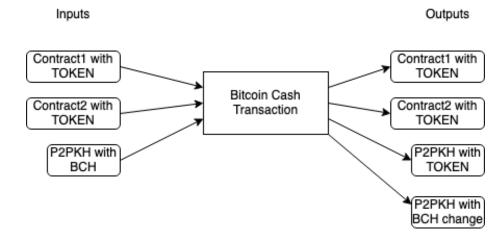


Figure 3: A token trade using two contracts for extra liquidity.

In a typical trade, multiple micro-pools are used as inputs along with additional

inputs to fund the trade. Contract UTXOs are recreated in corresponding output positions, and additional outputs send new funds back to the user. While funds are commonly held in P2PKH UTXOs, the contract does not mandate this.

Use Cases

The primary use case for the CPMM contract is to facilitate a decentralised exchange. Additionally, it can streamline merchant payments, allowing buyers to pay in tokens they don't hold by utilising the contract.

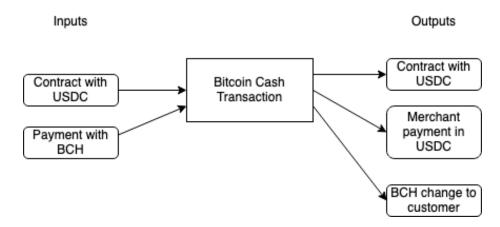


Figure 4: Paying a merchant in USDC, without the user owning any USDC.

For instance, a user with BCH can directly pay a merchant requesting USDC by engaging a Cauldron contract to perform the exchange within the transaction itself.

Conclusion

Cauldron's implementation of CPMM contracts on Bitcoin Cash leverages micropools and the Bitcoin Cash Script to efficiently facilitate decentralised exchanges and enable innovative applications in decentralised finance.

References

[1] D. Berenzon, "Constant Function Market Makers: DeFi's 'Zero to One' Innovation," Bollinger Investment Group, Apr. 23, 2020. [Online]. Available: https://medium.com/bollinger-investment-group/constant-function-marketmakers-defis-zero-to-one-innovation-968f77022159. [Accessed: June 20, 2023].

Acknowledgments

Thanks to Andrew Stone for introducing me to CPMM and suggesting its feasibility on the UTXO-based Nexa blockchain.

Thanks to Tom Harding for concept reviewing the contract.

Thanks to bitcoincashautist for reviewing and finding size optimizations in the contract.

Appendix - Cauldron contract

The following code in Bitcoin Cash Script makes up the code of the cauldron contract.

```
# If there is an input, it must be a pubkey and signature.
# This means that the owner wants to withdraw the contract.
OP_DEPTH,
OP_IF,
  # It's a withdrawal
      OP DUP,
      OP_HASH160,
      withdraw pkh,
      OP EQUALVERIFY,
      OP CHECKSIG,
OP_ELSE,
  # It's a trade
      # Verify it is the correct category ID.
      OP_INPUTINDEX,
      OP_OUTPUTTOKENCATEGORY,
      OP_INPUTINDEX,
      OP_UTXOTOKENCATEGORY,
      # Enforce version 2
      # Enforcing version is to make sure that tools that
      # use this contract stay compatible, when and if
      # transaction format changes in the future.
      OP_TXVERSION,
      2,
      OP_EQUALVERIFY,
      # Verify that this contract lives on on the output with
      # the same input as this contract.
      OP_INPUTINDEX,
      OP_OUTPUTBYTECODE,
      OP_INPUTINDEX,
      OP_UTXOBYTECODE,
```

```
OP_EQUALVERIFY,
     # Calculate target K
     OP_INPUTINDEX,
     OP_UTXOVALUE,
     OP_INPUTINDEX,
     OP_UTXOTOKENAMOUNT,
     OP_MUL,
     # On stack: K
     # Calculate fee for trade. Fee is ~0.3%.
     # (abs(bch out - bch in) * 3) / 1000
     OP_INPUTINDEX,
     OP_UTXOVALUE,
     OP_INPUTINDEX,
     OP_OUTPUTVALUE,
     OP_SUB,
     OP_ABS,
     З,
     OP_MUL,
     1000,
     OP_DIV,
     # On stack: BCH FEE, target K
     # Get effective output K when including the fee.
     OP_INPUTINDEX,
     OP_OUTPUTVALUE,
     # Subtract fee
     OP_SWAP,
     OP_SUB,
     OP_INPUTINDEX,
     OP_OUTPUTTOKENAMOUNT,
     OP_MUL,
     # Verify that effective K > target K
     OP_SWAP,
     OP_GREATERTHANOREQUAL,
OP_ENDIF
```