Hooks play an essential role in fee management, particularly for swapping and liquidity withdrawal. They can impose fees on both actions and allocate a portion to the hook contract. The swap fee can be either static or dynamically adjusted, with hooks having the authority to modify them based on a predefined algorithm if they are dynamic. However, withdrawal fees can only be collected and managed directly through the hook; there is no built-in provision within the pool for defining withdrawal fees. Additionally, hooks can allocate and distribute accrued fees to various stakeholders, including Swappers, Hook Creators, and Liquidity Providers. During pool creation, parameters such as fee type (static/dynamic), hook permissions for sharing swap fees, and authorization for withdrawal fees are set. Once established, these flags remain immutable, ensuring consistency and clarity in fee management.

No-op (No operation). In CrossFiGod, when a pool manager initiates a callback to a hook, the hook can respond with a specific byte4 selector to indicate a no-operation (No-op). This feature can serve various use cases; for example, it allows hooks to log transaction details for potential deferred execution or provides protection against front-running through strategic sequencing within the hook's logic.

3. Gas optimisations

The CrossFiGod Ecosystem implements a wide range of gas optimization techniques to enhance efficiency and reduce transaction costs. These optimizations include Singleton contract design, Flash Accounting with Transient Storage, compliance with the ERC-6909 multi-token standard, support for native gas tokens, and the exclusion of built-in features like price oracles. The following sections will provide in-depth explanations of these enhancements.

3.1 Singleton

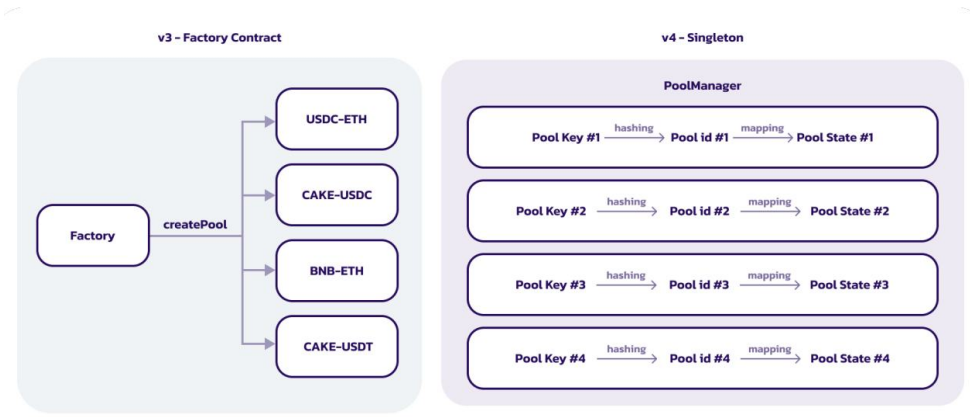

Figure 4: Singleton

In the CrossFiGod Ecosystem, the deployment of individual contracts for each liquidity pool previously led to increased gas costs for pool creation and multi-pool transactions. In contrast, CrossFiGod has adopted a Singleton contract approach, integrating all liquidity pools into a single contract.

This architectural enhancement eliminates the need to transfer tokens between different contracts during multi-hop transactions, resulting in significant reductions in gas expenses.

Within the Singleton contract, data for each liquidity pool is efficiently organized using a mapping structure. The poolId, generated by hashing and converting the PoolKey struct, uniquely identifies the state of each pool. The PoolKey struct encapsulates key parameters such as token addresses, fees, hook address, pool manager address, and other pool-specific settings like tick/bin spacing and hook flags, streamlining the creation and management of liquidity pools.

Moreover, CrossFiGod supports various pool types, with each type implementing its own independent Singleton pattern.

3.2 Flash Accounting

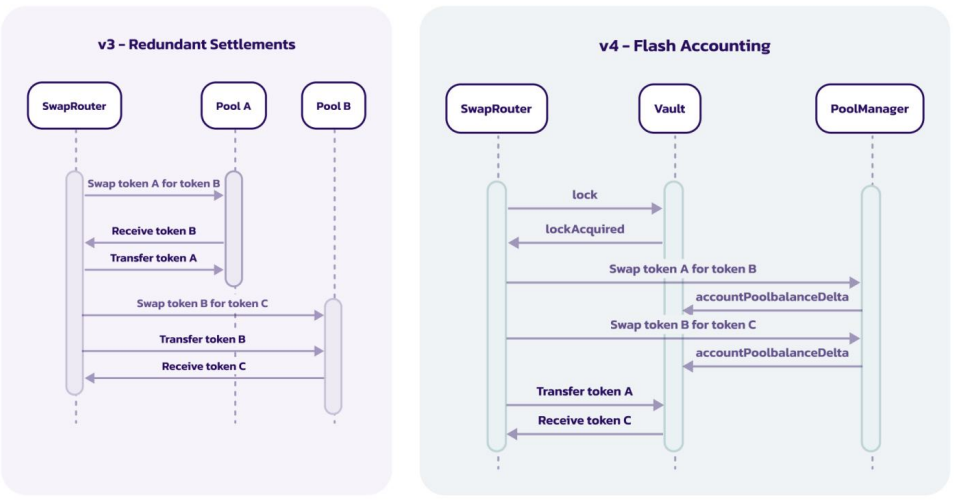

Figure 5: Flash Accounting

In the CrossFiGod Ecosystem, Flash Accounting optimizes the accounting process by calculating net balances for a batch of transactions and settling them collectively. This method significantly reduces gas consumption compared to the previous v3 model, where assets were transferred in and out of pools after each sub-transaction.

The optimization is made possible through a locking mechanism. When the lock is acquired, each operation updates an internal net balance until all transactions are processed. Once processing is complete, the lock is released to facilitate external transfers, thereby minimizing gas fees for multi-hop transactions.

Additionally, Flash Accounting leverages "Transient Storage," enabled by EIP-1153. This feature allows for temporary data storage that operates similarly to traditional storage but with a key difference: the values in transient storage are discarded after each transaction. This eliminates the need for serialization and deserialization processes associated with permanent storage, making transient storage operations more cost-effective and avoiding the high gas costs linked to read and write operations in permanent storage.

3.3 ERC-6909

In the CrossFiGod Ecosystem, the adoption of the ERC-6909 multi-token standard revolutionizes accounting by allowing users to store tokens directly within the Vault contract. This innovation eliminates unnecessary token transfers to and from the contract during typical swapping or liquidity management processes.

Traditionally, users would deposit or withdraw tokens from the vault after completing a transaction. However, with ERC-6909, if there is no immediate need to move tokens, users can mint an ERC-6909 token instead of performing a token transfer. In future transactions, they can simply burn the ERC-6909 token to settle their accounts, thus avoiding the need to transfer tokens back to the vault. This method can significantly lower gas costs, particularly for users who frequently engage in swapping and liquidity provisioning.

Moreover, ERC-6909 provides a streamlined alternative to the ERC-1155 standard by removing the need for callbacks and a single-operator permission model. As a result, ERC-6909 is recognized for its smaller code footprint and improved gas efficiency.

3.4 Support for Native Gas Token

With the introduction of Singleton and Flash Accounting, the CrossFiGod Ecosystem now enables the creation of liquidity pools using native gas tokens. This advancement allows for direct trading pairs with native gas tokens, removing the need for wrapping and unwrapping processes, which further reduces gas costs.

4 ONCHAIN LIQUIDITY MINING

The CrossFiGod Ecosystem implements a wide range of gas optimization techniques to enhance efficiency and reduce transaction costs. These optimizations include Singleton contract design, Flash Accounting with Transient Storage, compliance with the ERC-6909 multi-token standard, support for native gas tokens, and the exclusion of built-in features like price oracles. The following sections will provide in-depth explanations of these enhancements.

In the CrossFiGod Ecosystem, the deployment of individual contracts for each liquidity pool previously led to increased gas costs for pool creation and multi-pool transactions. In contrast, CrossFiGod has adopted a Singleton contract approach, integrating all liquidity pools into a single contract.

In the CrossFiGod Ecosystem, on-chain liquidity mining allows liquidity providers (LPs) to contribute liquidity and earn token rewards, as well as fees generated from their LP activities. Unlike PancakeSwap v3, where users must first add liquidity to receive an NFT and then stake it in the MasterChef to earn rewards, CrossFiGod simplifies this process. With CrossFiGod, LPs can start earning token rewards immediately upon providing liquidity, eliminating the staking step. This streamlined approach also enhances integration for DeFi protocols built on the CrossFiGod platform, enabling them to easily leverage the on-chain liquidity mining program.

REFERENCES

1] PancakeSwap v4 core. https://github.com/pancakeswap/pancake-v4-core/blob/main/docs/whitepaper-en.pdf

[2] Eip-1153: Transient storage opcodes. https://eips.ethereum.org/EIPS/eip-1153, 2022.

[3] AAVE Liquidity Book v1 - https://github.com/aave/aave-protocol/blob/master/docs/Aave_Protocol_Whitepaper_v1_0.pdf

[4] Erc-6909: Minimal multi-token interface. https://eips.ethereum.org/EIPS/eip-6909, 2023.

John Prophet
Crossfigod@gmail.com

CrossFiGod Ecosystem

September 2024

Abstract

The CrossFiGod Ecosystem is an innovative platform that implements a modular architecture for Automated Market Makers (AMM), promoting an adaptive and resilient design for decentralized exchanges. This architecture supports various AMM implementations, including CLAMM and Liquidity Book, with customizable hook contracts that allow for arbitrary logic to be executed at key points in the pool's operation. The modular design ensures seamless integration of new pricing curves, making the CrossFiGod Ecosystem flexible and ready for future changes.

Gas optimization techniques, such as Singleton contracts and Flash Accounting, help reduce gas costs, while the ERC-6909 standard enhances efficiency and lowers expenses for active users. Additionally, an advanced liquidity mining program on the platform incentivizes broader participation in the ecosystem, creating extra opportunities for users.

1 Introduction

The CrossFiGod platform is an innovative ecosystem designed from the ground up to optimize liquidity provision and trading on decentralized markets. We implement a Concentrated Liquidity Automated Market Maker (CLAMM) model, allowing users to provide liquidity within specific price ranges, significantly enhancing capital efficiency and deepening liquidity for traders.

Key Features:

1. Adaptive Architecture: Our platform separates accounting logic from automated market-making logic, allowing for easy adaptation to new paradigms and functionalities without the need to rewrite the core structure.

2. Customization Flexibility: Pool creators have the ability to customize their liquidity pools through hook integration. This allows for the addition of unique features such as custom oracles, dynamic fees to reduce impermanent losses, and active liquidity management strategies.

3. Gas Cost Optimization: We implement an independent Singleton implementation for each AMM logic and pool type, optimizing gas costs when creating pools and conducting multi-network transactions.

4. Flash Accounting: This feature consolidates transaction calculations, significantly reducing gas consumption by calculating net balances for a group of transactions.

CrossFiGod aims to create a user-friendly interface, minimizing the complexities of interacting with the platform while ensuring a high degree of security and transparency.

2 Arhitecture

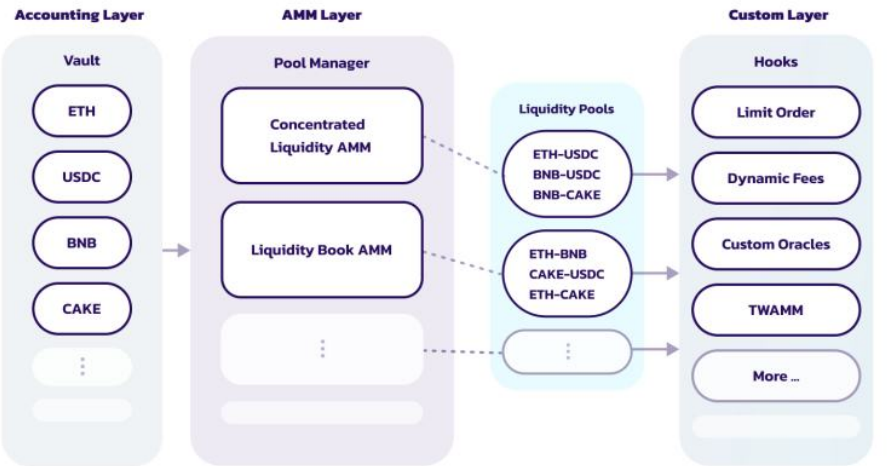

Figure 1: Three-Layered Architecture

Implements a three-layered modular architecture that consists of the Vault, Pool Managers, and Hooks. This structured approach ensures a high degree of flexibility and scalability within the platform

2.1 Vault

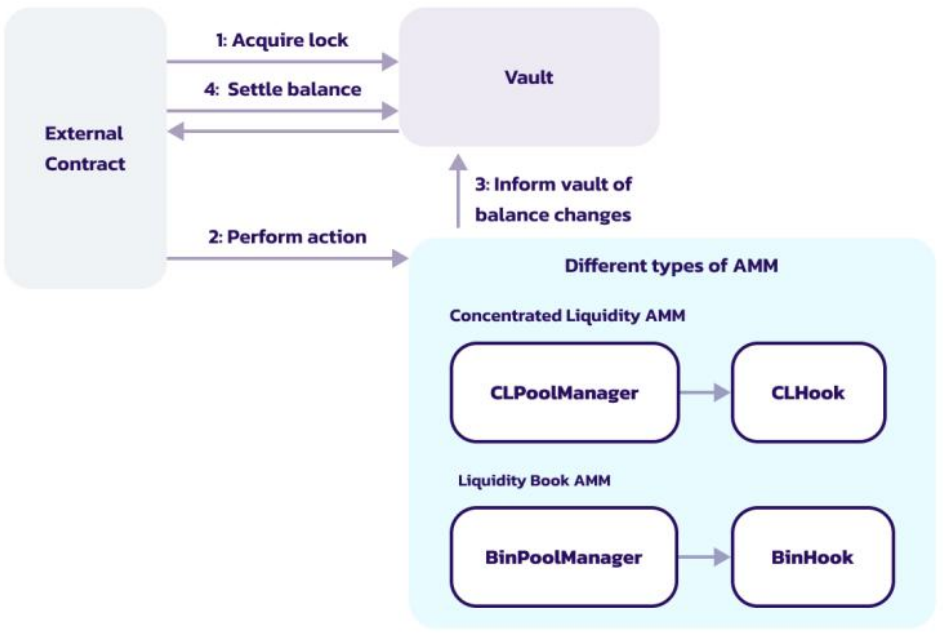

Figure 2: Vault Mechanism

At the core of the CrossFiGod Ecosystem is the Vault, which serves as an immutable accounting layer. It maintains a ledger of tokens that are either deposited or owed, facilitating a secure and efficient settlement process at the end of each transaction. The standard procedure for a transaction involves the following steps:

1. Acquiring a lock from the Vault
2. Executing operations such as swaps or adjusting liquidity levels
3. Updating the Vault with balance changes for each operation by the pool manager
4. Reconciling the net balance with the Vault, involving either depositing into or withdrawing tokens from the Vault

This method offers substantial savings on gas fees through the implementation of Flash Accounting with Transient Storage, particularly evident in transactions involving multiple pools. For instance, in multi-hop swaps like A→B→C, it's typically required to transfer all three tokens A, B, and C between the pools for A→B and B→C swaps. However, with the CrossFiGod architecture, since token B's balance is offset during the swap, only tokens A and C need to be moved, significantly streamlining the process.

2.2 Pool Manager

Complementing the Vault are various Pool Managers, each a singleton contract that encapsulates the logic for different types of Automated Market Makers (AMMs) within the CrossFiGod Ecosystem. These include, but are not limited to, the CLPoolManager and BinPoolManager.

The CLPoolManager implements the CLAMM model, which is well-known for its widespread adoption and popularity. It operates on the principle of constant product invariance. In contrast, the BinPoolManager is based on the Liquidity Book AMM model, introduced by Trader Joe. This manager utilizes the constant sum invariant, providing a unique approach to liquidity management.

The AMM layer facilitates the seamless integration of various pool types within the CrossFiGod Ecosystem. When a new PoolManager contract is developed, it is up to the governance body to vote on whether to incorporate this new manager into the ecosystem.

2.3 Hooks

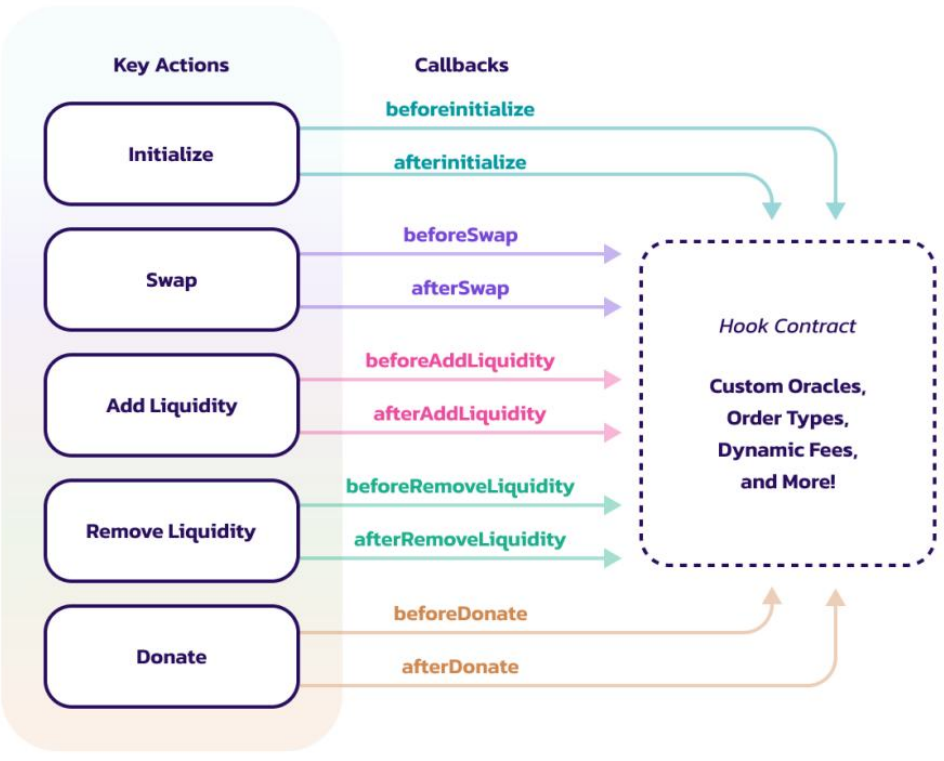

Figure 3: Hook Callbacks

Hooks are externally deployed contracts that operate independently from the core AMM logic and Pool Managers within the CrossFiGod Ecosystem. These contracts can be deployed by anyone, including developers and protocols, and execute predefined logic during key actions in a pool's operation. They function like widget-like add-ons, enhancing liquidity pools with additional functionalities and features.

When initializing a pool, the creator has the option to specify a hook contract associated with that pool. Key actions include 'initialize', 'swap', 'addLiquidity', 'removeLiquidity', and 'donate'. Hooks empower creators to integrate custom logic before or after these actions, enabling a wide array of use cases.

CrossFiGod supports hook callbacks at several specific points: beforeInitialize / afterInitialize, beforeAddLiquidity / afterAddLiquidity, beforeRemoveLiquidity / afterRemoveLiquidity, beforeSwap / afterSwap, and beforeDonate / afterDonate. These points provide granular control over when hook logic is executed, allowing for precise customization of pool behavior.