🔀

HyPerps

For contract implementation, see: https://github.com/eason-w/HyPerps/tree/main/contracts

Overview

HyPerps is a cross chain perpetual futures protocol built on top of Hyperlane. HyPerps will allow users to deposit collateral on one chain, then trade perpetual futures on another chain. HyPerps uses an oracle based model, allowing users to open and close positions at any time with the price execution according to the oracle feed. Traders trade against the liquidity of HyPerps, which can live on any chain that HyPerps is deployed on.

▼ Example

A user can deposit 1000 USDC into HyPerps on Chain A. The user can then go on Chain B to open, close, and manage their HyPerps perpetual futures positions using the collateral from Chain A.

HyPerps solve a number of problems and limitations of existing perpetual futures protocols:

Gas fees - trading perps on L1s can be extremely gas intensive. Having the ability for the collateral to live on one chain whilst executing trades on another allows users to save money on gas. This also allows traders to save fees from bridging the collateral to another chain

▼ Example

A user has the majority of their money on Chain A, and regularly has $10 gas fees to send or deposit tokens. The user does not have much money on Chain B, but gas fees are regularly $0.01. On HyPerps, this user would be able to pay a one-time gas fee to deposit their collateral on Chain A while executing all of their trades on Chain B, all without bridging any funds

Latency - some chains publish multiple blocks in a second while others only publish a few blocks per minute. Trading perps on a high latency chain can lead to poor user experience in terms of trade execution and price updates. HyPerps gives the ability to users to use a faster latency chain for better trade execution

▼ Example

Chain A has 15 second block times. Chain B has 1 second block times. User wants to trade on Chain B for faster trade execution but only has assets on Chain A. User can use HyPerps to trade perps on a higher latency chain.

Liquidity - L1s almost always have more liquidity & assets than L2s, while L2s have lower latency and lower gas fees. By unifying and utilizing liquidity on other chains, it allows traders to execute larger positions that require more liquidity, even if the chain they are trading from has low liquidity

▼ Example

Chain A has 10b dollars in potential liquidity and 10m in HyPerps, Chain B has 100m of potential liquidity and 100k in HyPerps. Traders executing on Chain B have access to 1.1m of liquidity, whereas traditional platforms would only allow traders on Chain B to have access to 100k of liquidity.

Asset type limitations - some tokens are not available on certain chains, or have extremely low liquidity. HyPerps allows traders to open positions on assets of other chains, removing limitations of only having native asset types to trade with on a given chain.

▼ Example

Chain A’s native asset, $A, is only available on Chain A. A user on Chain B would like to trade $A without using Chain A. The user could use HyPerps to open a perps position on $A while on Chain B.

Protocol Architecture

Protocol Liquidity

Users can deposit tokens into the HyPerps liquidity pool to provide liquidity to the perps platform. This is accepted on any supported chain. Initial accepted collateral types to the liquidity pool on are USDC, wETH, and wBTC. These tokens are the tokens that users will be able to trade on the platform, and are also the acceptable collateral types. The liquidity pool acts as an index, with users getting credited with their share of the pool when they deposit.

Price Oracle

HyPerps uses Pyth for the price oracle which determines opening price, closing price, PnL, and whether or not positions are eligible for liquidation. Pyth price feeds are queried each time a relevant action is performed, and the price message is sent back to the collateral chain of the user.

Depositing Collateral

Users can deposit collateral on any chain that has liquidity. Accepted collateral types are any tokens in the HyPerps liquidity pool (USDC, wETH, wBTC). This collateral will be the collateral of the user's position when they trade on another chain. This credited collateral will remain in the contract and can increase with additional deposits or profitable trades, and decrease with withdrawals and unprofitable trades.

Opening a Position

Opening a perps position can occur on any chain, even the chain where the collateral lies. The inputs to the openPosition() function are: `assetType`, `collateralType`, `collateralSize`, and `leverage`. The function also calculates `positionSize` as well as call Pyth to determine the `openingPrice` and `liquidationPrice`. On open, the position struct variable `isOpen` is set to `true`. Longing an asset requires the collateral to be of that asset, while shorting requires the collateral to be in USDC. The maximum `leverage` that a position can be opened at is 10x, with liquidations occurring at 50x leverage. The positionSize is calculated as: `positionSize` = `collateralSize` * `leverage`.

Closing a Position

Similar to opening, a perps position can be closed on any chain. The inputs to the closePosition() function are: `positionIndex` & `collateralChain`. The function will also call Pyth to determine `closingPrice` and `PnL`. This function also sets `isOpen` to `false` to avoid additional actions to the protocol. These values are then sent to the collateralChain to settle the position. PnL is calculated by: ((`closingPrice` / `openingPrice` -1)* `leverage`)* `collateralSize`. This PnL is then apllied to the user's collateral.

Liquidations

Liquidators can call the liquidate() function on any chain, but must specify the destination chain where the collateral of the position lies, as well as the index of the position. Positions are only eligible to be liquidated if the assetPrice passes the liquidationPrice (greater than for shorts, less than for longs). liquidationPrice is calculated as: `liquidationPrice` = openingPrice*(openingPrice/leverage*0.95). A 5% buffer is enforced to ensure no bad debt is created and to incentivize liquidators.

▼ Example scenario of a user using HyPerps

Bob keeps most of his funds on Ethereum mainnet and has a little bit of gas on Arbitrum. Bob wants to trade ETH perps but does not want to use an Ethereum perps exchange due to high gas fees and high block latency, however, they also do not want to bridge to Arbitrum.

Bob decides to use HyPerps, getting the best of both worlds. Bob sends a deposit of 1 ETH on HyPerps' Ethereum deployment. This collateral deposit message is propagated to all other chains that HyPerps is deployed to. Bob then goes to Arbitrum to manage his position. He opens an ETH long with 10x leverage using his 1 ETH as collateral. The opening price at the time of the transaction is $1000. This open position message is sent and received on Ethereum, where the 10 ETH of liquidity in the LP is reserved. ETH price is now $1100 so Bob decides to close his position. This message is sent to the Ethereum deployment. Bob's PnL would be (($1100 / $1000 -1)* 10)*1 ETH = 1 ETH. Bob earns 1 ETH from this trade and now has 2 ETH of collateral on Ethereum mainnet.

▼ ETHWaterloo Implementation

For HyPerps' MVP at ETHWaterloo, HyPerps has been deployed on Ethereum Goerli, Arbitrum Goerli, Polygon zkEVM Testnet, and Gnosis Mainnet. The current implementation on Ethereum Goerli <> Arbitrum Goerli is the most recent and stable.

Certain compromises had to be made in the sake of time, including the implementation of a hub and spoke model, with liquidity and collateral residing on the hub with ability for trade orders to be placed on any chain.

The Pyth price feed is not being used in the test implementations to ensure all functions are working properly (ex. liquidations). Prices are manually being set to test the protocol

Relevant contracts & GitHub:

- Main GitHub page
  - https://github.com/eason-w/HyPerps/tree/main
- HyPerpsHub contract
  - https://github.com/eason-w/HyPerps/blob/main/contracts/HyPerpsHub.sol
- HyPerpsSpoke contract
  - https://github.com/eason-w/HyPerps/blob/main/contracts/HyPerpsSpoke.sol
- SingleChainPerpsProtocol contract
  - https://github.com/eason-w/HyPerps/blob/main/contracts/SingleChainPerpsProtocol.sol