Deep Dive on DeFi Lending Protocols

Fanglin Lu
04/18/2021

Compound Finance

- TVL: $10.94B (04/18/0201, from defipulse.com)
- Compound Protocol White paper
- Compound Cash / Compound gateway: an open, distributed ledger for cross-chain interest rate markets.
    - In testnet now (04/24/2021)
- Developer doc
- Github
- Coding language: solidity
- Chains support: ETH
- Governance token: COMP
- Contracts on Compound (code link):
    - *CToken (code), CErc20 (code) and CEther (code)*
      The Compound cTokens, which are self-contained borrowing and lending contracts. CToken contains the core logic and CErc20 and CEther add public interfaces for Erc20 tokens and ether, respectively. Each CToken is assigned an interest rate and risk model (see InterestRateModel and Comptroller sections), and allows accounts to *mint* (supply capital), *redeem* (withdraw capital), *borrow* and *repay a borrow*. Each CToken is an ERC-20 compliant token where balances represent ownership of the market.

      Note here that CEther contract and CErc20 contract have different logic. When supplying Ether to the Compound protocol, an application can send ETH directly to the payable **mint** function in the cEther contract. Following that mint, cEther is minted for the wallet or contract that invoked the mint function. Remember that if you are calling this function from another smart contract, that contract needs a **payable** function in order to receive ETH when you redeem the cTokens later. The operation is slightly different for cERC20 tokens. In order to mint cERC20 tokens, the invoking wallet or contract needs to first call the **approve** function on the **underlying token's contract**. All ERC20 token contracts have an **approve** function.

    - *Comptroller*
      The risk model contract, which validates permissible user actions and disallows actions if they do not fit certain risk parameters. For instance, the Comptroller enforces that each borrowing user must maintain a sufficient collateral balance

across all cTokens.

    - *Comp (code)*
      The Compound Governance Token (COMP). Holders of this token have the ability to govern the protocol via the governor contract.

    - *Governor Alpha*
      The administrator of the Compound timelock contract. Holders of Comp token may create and vote on proposals which will be queued into the Compound timelock and then have effects on Compound cToken and Comptroller contracts. This contract may be replaced in the future with a beta version.

    - *InterestRateModel*
      Contracts which define interest rate models. These models algorithmically determine interest rates based on the current utilization of a given market (that is, how much of the supplied assets are liquid versus borrowed).

    - *Careful Math*
      Library for safe math operations.
    - *ErrorReporter*
      Library for tracking error codes and failure conditions.

    - *Exponential*
      Library for handling fixed-point decimal numbers.

    - *SafeToken*
      Library for safely handling Erc20 interaction.

    - *WhitePaperInterestRateModel (code)*
      Initial interest rate model, as defined in the Whitepaper. This contract accepts a base rate and slope parameter in its constructor.

- Codebase architecture:
    - Network
        - This folder contains the configuration for given networks (e.g. `rinkeby.json` is the configuration for the Rinkeby test-net). These configuration files are meant to be used to configure external applications (like dApps) and thus contain a base set of information that may be useful (such as the address the Comptroller and a list of cToken markets). These configuration files are auto-generated **(Question: how are they auto-generated?)** when doing local development.
    - Contract
        - This is where all the contracts on Compound are defined
    - Script

        - Saddle
        - Scen

Cream Finance

- Total Supply, total borrow (04/18/2021, from cream website):
    - Ethereum: $326,528,744.54, $79,231,528.97
    - Iron Bank: $492,144,663.95, $223,155,159.83
    - BSC: $122,220,887.36, $30,176,450.28
    - Fantom: $140,270.78, $40,093.72
- Resource link
- Github
- Chains support: ethereum, iron bank, binance smart chain and fantam
- A lending platform based on Compound Finance and an exchange platform based on Balancer Labs
- Support more tokens than Compound Finance
- Forked from Compound (comparison between compound code and cream code)

Venus Protocol

- TVL: $10,195,804,338.65 (04/18/2021, from Venus website)
- Github
- Chains support: BSC
- Stable coin: VAI
- Governance token: XVS, SXP

Aave

- Github
- Chains support: ETH, Polygon

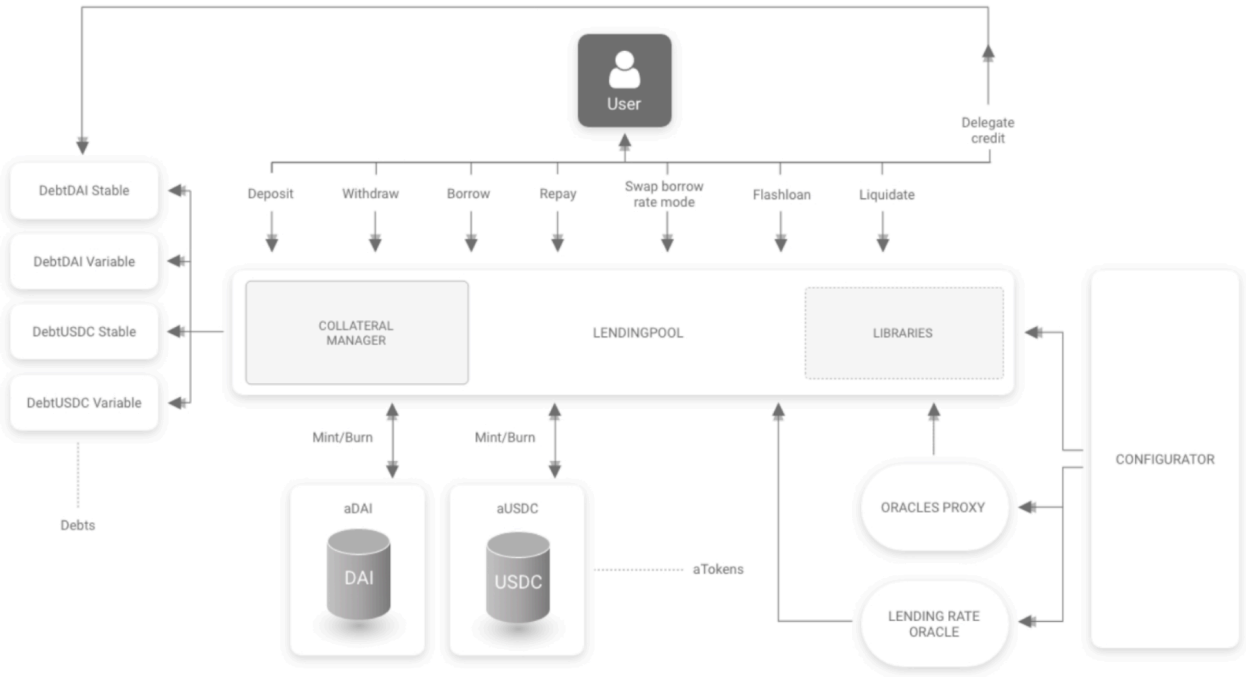

Figure1: Aave V1 Protocol Architecture

- Main Contracts:

- **LendingPool** (documentation link)
  The main entry point into the Aave Protocol. Most interactions with Aave will happen via the LendingPool.
  - Methods:
    - deposit()
    - withdraw()
    - borrow()
    - repay()
    - swapBorrowRateMode(): swapping between stable rate or variable rate
    - setUserUseReserveAsCollateral():
    - liquidationCall()
    - flashLoan() (If we don't want flashLoan, we can just comment out this method)

  Question: Why do we need at least an allowance() of amount of the assets

  > When depositing, the `LendingPool` contract must have at least an `allowance()` of `amount` for the `asset` being deposited. This can be done via the standard ERC20 `approve()` method.

  being deposited in the following documentation?

- **LendingPoolAddressesProvider**
  The main addresses register of the protocol, for particular markets. The latest contract addresses should be retrieved from this contract by making the appropriate calls.
- **LendingPoolAddressesProviderRegistry**
  Contains a list of active LendingPoolAddressesProvider addresses, for different markets.
- (TODO)

- Features in Aave V2:
  - Credit Delegation
    Native Credit Delegation (**CD**) is a new feature in Aave v2. It allows a depositor to deposit funds in the protocol to earn interest, and delegate borrowing power (i.e. their credit) to other users. The enforcement of the loan and its terms are agreed upon between the depositor and borrowers, which can be either off-chain via legal agreements or on-chain via smart contracts.

Kashi

- Kashi, a lending and margin trading platform, is the first application within BentoBox. On the lending side, Kashi's core innovation revolves around isolated lending pairs which aim to tackle a limitation of popular lending platforms. One key drawback of protocols like Compound and Aave is that the entire platform is exposed to the risks of individual assets. Every user and every asset is equally affected if the protocol becomes under-collateralized, caused by a shock to the market price of collateral assets. Isolated lending pairs seek to improve on this by allowing anyone to create lending pools using any pair of tokens. Depending on the tokens used, some pools will be highly volatile while others will be more stable, but the risk will not spread throughout the platform due to the pools' isolated nature. The mechanism will enable creating a wide range of lending pairs which will result in an increased set of shorting options for traders. This feature is in high demand but is currently not possible due to the limited variety of tokens available for borrowing.
- Link

Some Questions for building the Harmony lending protocol

Stable coin

We can easily use stablecoins brought over the bridge. We have BUSD, and then technically we can also use USDT/USDC (HRC20)

Execution Plan

Aave Contracts:

- Price Oracle deployment
    - Replace chainlink price oracle with band protocol price oracle (Estimate 5-6 days)
- Addresses Provider deployment (Estimate 1-2 hours)
- Addresses provider registry deployment (Estimate 1-2 hours)
- Lending pool deployment (Estimate 5-6 hours)
- Aave price oracle and lending rate oracle deployment (Estimate 4-5 hours)
- AAVE Data provider deployment (estimate 1-2 hours)
- WETH(WONE on harmony) gateway deployment (estimate 1 hour)

Estimation: 8-10 days

Testnet contract addresses:
LendingPoolAddressesProvider: 0x1b8A438aed094379740810690d5c15E7216D80E3
LendingPoolAddressesProviderRegistry: 0x2aFdAD94BA714B2474735778c38F2745b99261Bd
ReserveLogic: 0x285966890554c3B7b28A574513d7EACeaed1eC73
GenericLogic: 0xb24B7ba8Ec6E9F946622D3635137aC483b53Dc51
ValidationLogic: 0xD01ab44302f83F2f31e572aE38d9B6A121428253
LendingPoolImpl: 0xE769dAF8c7baF13b46325fb625A91C3452bB29DA
LendingPool: 0xCeCadE29d8018e10A76BE22037efBE0bf3217AF6
LendingPoolConfiguratorImpl: 0x7243e9B082FFC494e8E9DA1eE0f93748A431D65D
LendingPoolConfigurator: 0x09C771B9Af30B5F360F5945D615A727d3B2E08ac
StableAndVariableTokensHelper: 0x44c09a12A687046C1F6377003F15A7B68B5BdD31
ATokensAndRatesHelper: 0x756EB321A404912b1C9135C8f9cf01489BD56D54
AaveOracle: 0x739FfEb824297B506FD116Eab8c8f688bF88FbE3
LendingRateOracle: 0x359D9282258385E6aAFd8066525E34B0744dB98C
AaveProtocolDataProvider: 0x264F97A173098d8C65D4487F0cF546f015A4183C
stableDebtDAI: 0x308282601c728EAe61348441406A2602CB4D4D89
variableDebtDAI: 0x1666356858fc344C25f4935a90267Aa6Dd2c3f10
aDAI: 0x27F9738fdC720C70B8e875bA5ec7f8171C7Aa8dF
strategyDAI: 0xd004fC7781B7889F2e913785700c71E5c3A536e6
stableDebtUSDC: 0x18ed1914c71b98D2cBC2BcB50A744085BBaB70e0
variableDebtUSDC: 0x5Eed92e9a750F7f167625621078Bf3fcDdbe2117
aUSDC: 0x6ff935cad761a0627Ef873FC719834f6321ab984
strategyUSDC: 0xc46cDbaFfD6A8a8ba2F31fd598C93b4CADf4813C
stableDebtUSDT: 0x5Ebb99F1fDd5d35F28164Bec8f90ED98C7b5Ef4f
variableDebtUSDT: 0x972b9aDB04443ba34b26A3C6dF4262aecFa115Cd
aUSDT: 0x778dCF83145BED738148fef00fE163EEcc4e49E2
strategyUSDT: 0xaEEd91a0B84DA35A7069abb0086bB747F637a471
LendingPoolCollateralManagerImpl: 0x21bA7819A24f6c00acD3A453D0aAAD03801c6c69
LendingPoolCollateralManager: 0x21bA7819A24f6c00acD3A453D0aAAD03801c6c69
WalletBalanceProvider: 0xc906ecC98134Ba166C40b34983FDD9C8CB769222
WONEGateway: 0x599f7729B6E4b4Bc9D330899da7f59D541EaaE4c
stableDebtWONE: 0x766443FF93A122caB68C6c9FaD1280EEe103EFDA
variableDebtWONE: 0x054969fA2f02fdC449563226818367D963824F3f
aWONE: 0xED4B3433E9f674b7DBcb6Da030ec097Dd3485882
strategyWONE: 0xa13788C37BFe5883aaCe1215e987579477c0a397

Deposit coins(Testnet):
DAI: 0x14e1AF20a14d8357AcF523DE56FB1D9B16F6179e
USDC: 0x9bC9937Ad5010403a127BCD41ff2F7036ccB9559
USDT: 0xB5353E162F27BB5Fa81eFAeB09bECfFE49FCf215
WONE: 0xBbD9C6ba46F0e03AA4098c2397B2b21c77894684