WHITE BOOK

SOSODEFI

DEVELOPER

August 2020 | Version 1.0

CATALOG

Introduction 3

1.System Structure of SosoDefi 4

2. Loan Strategy 5

- 2.1 Loan pool 5
- 2.2 Interest rate strategy 5
- 2.3 Fixed interest rate strategy 6

3 Model PIPELINE 8

- 3.1 The principle PIPELINE 8

4. Oracle 10

- 4.1 The principle of oracle 11
- 4.2 The application of oracle in SosoDefi 12

5. Option pricing model 13

- 5.1 B-S Model 13
- 5.2 B-S Pricing Formula 14

6. Yield Farming 15

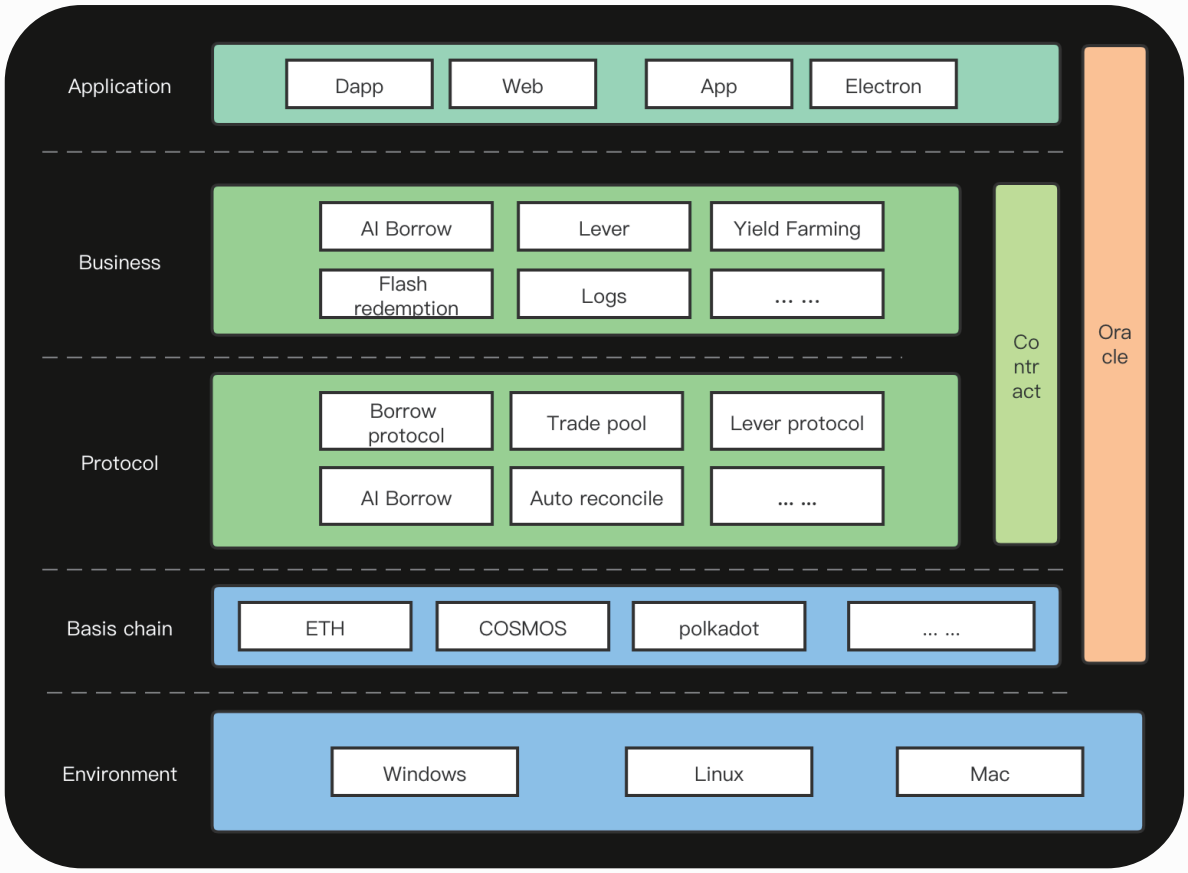
- 6.1 The principle of yield farming 15
- 6.2 The profit mode of yield farming 16

7. Conclusion 17

INTRODUCTION

The SosoDefi protocol is a pool-based borrowing strategy. The lender puts the Token into the pool smart contract to provide liquidity. At the same time, collateral can be placed to borrow funds. Loans do not need to be matched separately, but rely on the funds in the pool smart contract, the borrowed amount and the mortgaged Token.

SYSTEM STRUCTURE OF SOSODEFI



As shown in the figure above, SosoDefi has the following components:

Application layer: We will provide our own Dapp, App, and desktop client.

Business layer: The business layer contains all the functional algorithms and calculation processes required in the system, and implements the protocol layer in detail.

• Calculate the token equivalent of the user's balance (debit balance, mortgage balance, and liquidity balance) to evaluate how much the user is allowed to borrow and the health factor.

• Summarize the funds in the loan pool and provide them to the borrower.

• Calculate the loan interest rate and find the optimal interest rate.

Protocol layer: Provide an abstract model for developers to make expansions.

LOAN STRATEGY

2.2 Loan pool

The following operations are provided:

【Deposit, redemption, borrowing, repayment, liquidation, fast loan】

One of the functions of the smart contract loan pool is tokenization. When a user deposits a specific Token, he will receive a corresponding number of sTokens, which are used to generate liquidity mapping.

Interest on underlying assets. sTokens are generated when user deposit his assets, and its value continues to increase until redemption or liquidation is triggered. Whenever a user triggers a borrowing operation, the Token used as a mortgage will be locked and cannot be transferred.

2.2 Interest Rate Strategy

The "Interest Rate Strategy" smart contract holds the information required to update the interest rate of the specific reserve and performs the updating of the interest rate . Each reserve has a specific smart contract of "interest rate strategy", especially in the basic strategic contract.

Now we will define the following concepts:

• Basic floating interest rate (Rv0)

• Slope of interest rate which is below the optimal utilization rate (Rs1)

• Slope of interest rates which is above the optimal utilization (Rs2)

The current floating borrowing interest rate is calculated as:

$$if U < Uoptimal, Rv0 = Rv0 + \frac{U}{Uoptimal} Rs1$$
$$if U \geq Uoptimal, Rv0 = Rv0 + Rs1 + \frac{U - Uoptimal}{1 - Uoptimal} Rs2$$

This interest rate model allows calibration of key interest rates：

- $At[U = 0], Rv = Rv0$
- $At[U = Uoptimal], Rv = Rv0 + Rs1$

The Uoptimal above has considered the capital cost and the sharp rise of the interest rate.

2.3 Fixed interest rate strategy

The last and most important constraint of the stable interest rate model is interest rate rebalancing. This is to solve the problem of changes in market conditions or increased Token costs in the pool.

Stable interest rate rebalancing will happen under two specific circumstances:

1.Rebalancing up. When the user earns interest through lending, the fixed interest rate (x) is equal to the latest rebalanced value Bs.

$$B[x, s] < RI$$

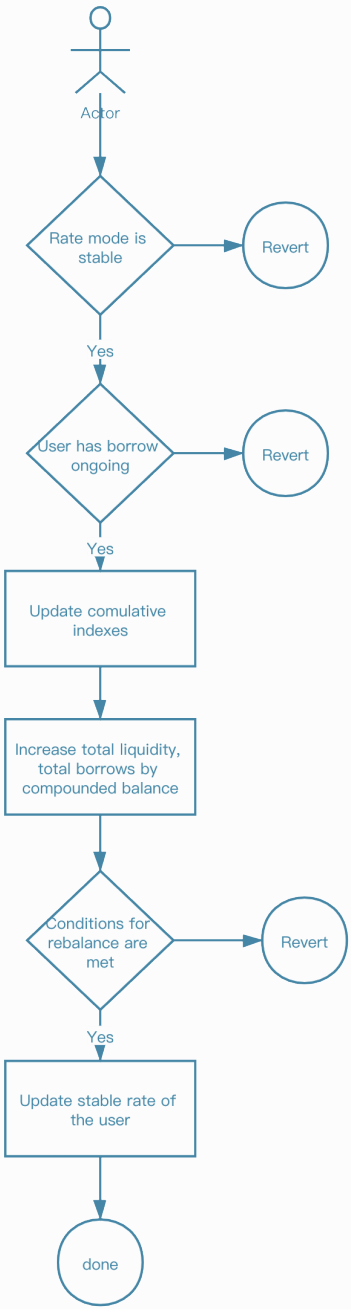
2.Rebalancing down. The fixed interest rate (x) is equal to the latest rebalanced value Bs.

$$\text{if: } B[x, s] > Bs(1 + Bs)$$

The function of loan contract opening rebalances the stable borrowing rate, which can rebalance the stable interest rate interest of specific users, and anyone can call this function. However, there is no direct incentive for callers to rebalance rates for specific users. Therefore, SosoDefi will provide an agent that regularly monitors all stable interest rate positions and rebalance those positions deemed necessary. The rebalancing strategy will be determined by the agent, which means that users who meet the rebalancing conditions may not rebalance immediately.

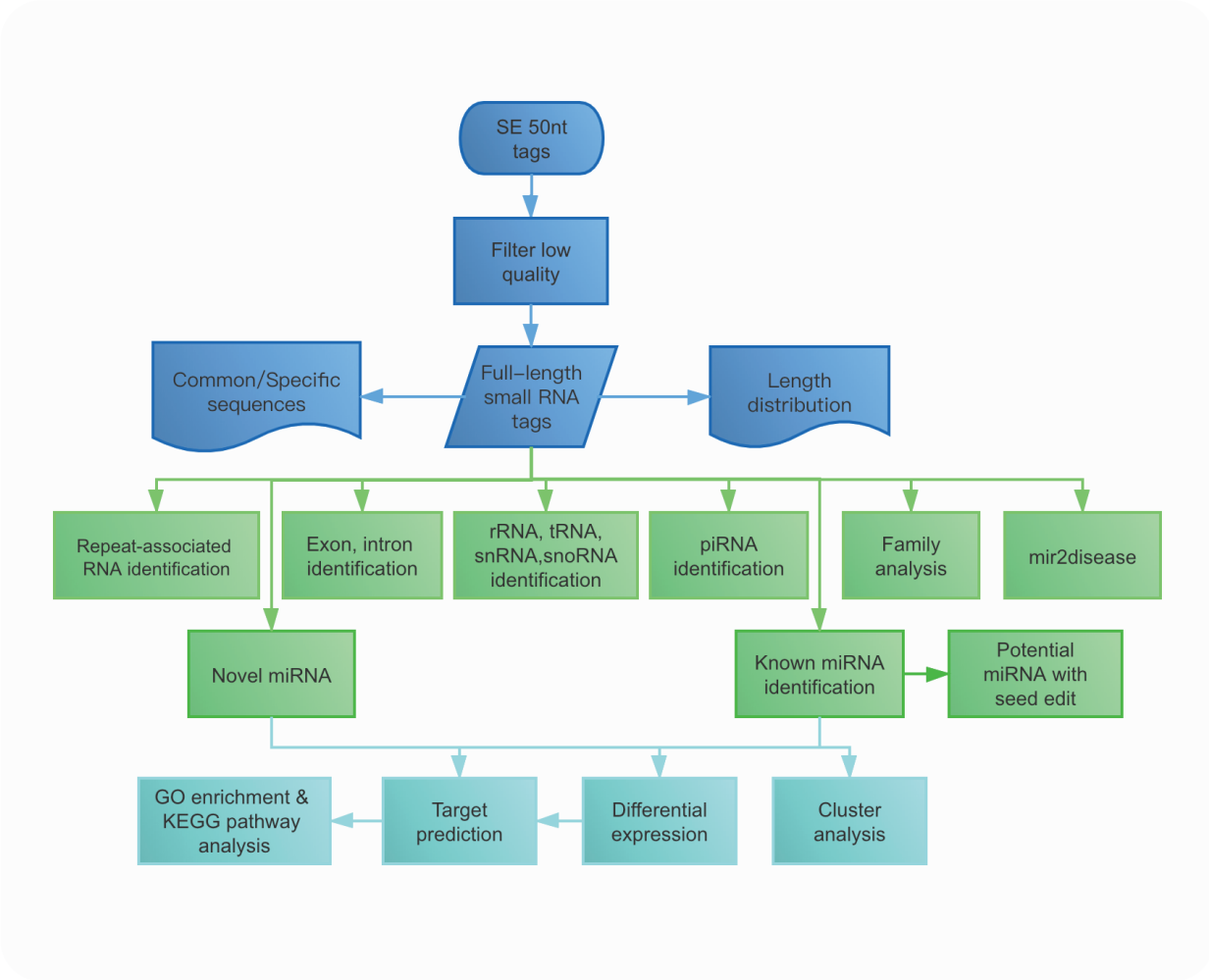
Since these conditions depend on the available liquidity and market conditions. Therefore, there is no need for immediate rebalancing in some temporary situations.

This will not add any centralization element to the agreement. Even if the agent stops working, anyone can call the agent's rebalancing function. Although there is no direct reason for this, it has an indirect impact on the ecosystem. In fact, even if the agent ceases to exist, the depositor may still want to trigger a rebalancing of the minimum borrowing interest rate position to increase the liquidity ratio and/or force the borrower to liquidate, thereby increasing available liquidity. On the contrary, if the scale is reduced, borrowers will directly incentivize them to adjust their positions to reduce interest rates.The following flowchart explains the operation sequence of this function. The accumulated compound interest balance is not affected by the rebalancing until the moment of rebalancing.



MODEL PIPELINE

3.1 The Working Pipeline



Suppose you want to provide liquidity in Uniswap's sETH pool and earn transaction fees and SNX rewards.

You notice that Synthetix has just started distributing SNX rewards to Curve sUSD liquidity providers.

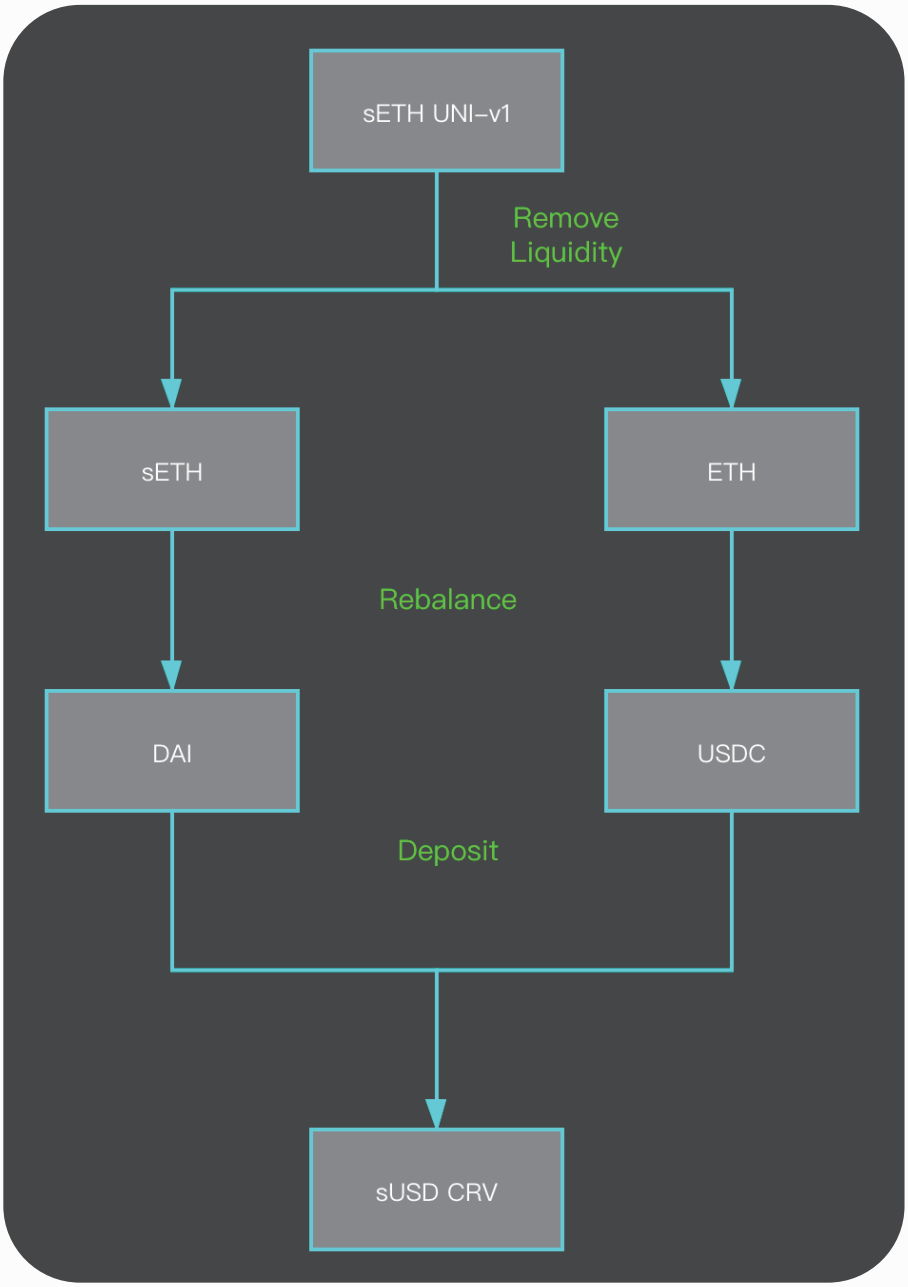
To transfer part of the liquidity assets from the Uniswap sETH pool to the Curve sUSD pool, you need to perform at least 4 chain transactions:

1. Remove liquidity from Uniswap's sETH pool

2. Use some ETH to buy DAI

3. Use sETH to buy USDC

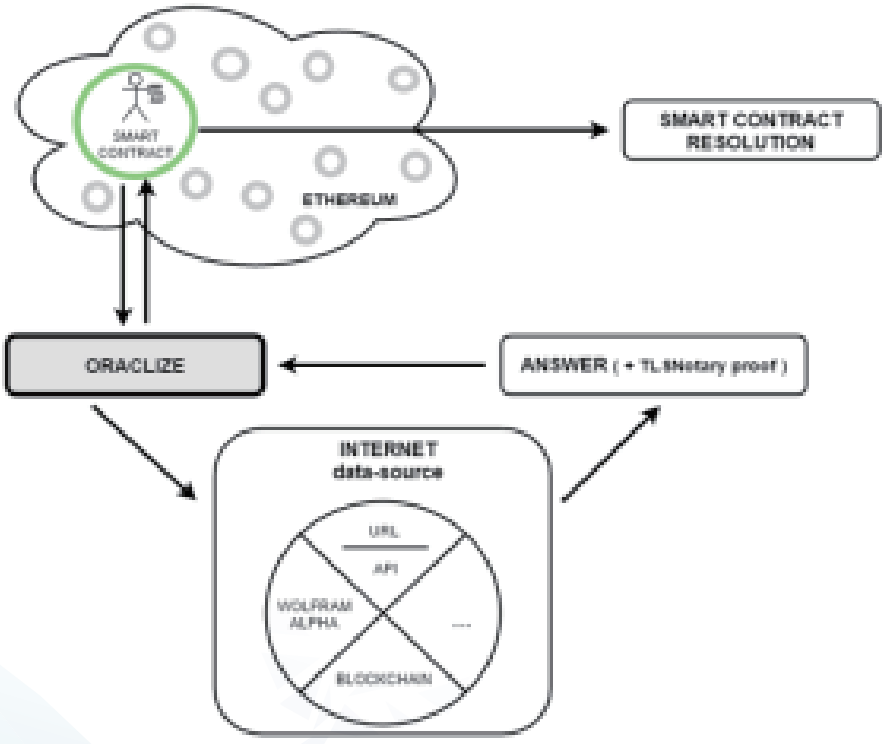
4. Add DAI+USDC in Curve



Blockchain is a transaction-based state machine. What it can do is very simple, that is transforming the blockchain from a state Into another state by submitting transactions to the blockchain. In order to maintain the consensus, the execution process of the VM must be completely determined and based only on the shared context of the Ethereum state and the signed transaction. This has two particularly important consequences: one is that VMs and smart contracts have no inherent source of randomness; the other is that external data can only be introduced as the data load of the transaction. In layman's terms, the blockchain does not have the ability to obtain data actively, it can only use the data of the blockchain itself. The lack of data leads to a very small application range of smart contracts. At present, most applications are developed around tokens.

Therefore, if we want the contract to have more capabilities, an oracle needs to be introduced.

4.1 The principle of oracle



In the blockchain field, an oracle is considered a system that can provide external data sources for smart contracts. From the perspective of traditional technical architecture, the oracle is the middleware that connects the smart contract with the outside world of the blockchain, which is an important infrastructure of the blockchain. Its role is to provide data for the smart contract on the blockchain.

4.2 The application of oracle in SosoDefi

The designing modes of oracle are:

• immediate-read

• publish–subscribe

• request–response

```
function requestEthereumPrice(address _oracle, string _jobId)
  public
  onlyOwner
{
  Chainlink.Request memory req = buildChainlinkRequest(stringToBytes32(_jobId), this, this.fulfillEthereumPrice.selector);
  req.add("get", "https://oracle-api.sosodefi.com/data/gas-price");
  req.add("path", "Price");
  req.addInt("times", 100);
  sendChainlinkRequestTo(_oracle, req, ORACLE_PAYMENT);
}
```

SosoDeFi uses oracle models such as chainlink to estimate the gas fee, and calculates the corresponding interest rate of each current bag, and adjusts then to the optimal distribution ratio.

Its function is to obtain the gas price of ETH from the specified API (SosoDefi). The parameters passed by the function are the specified Oracle address and jobId. After assembling the listed request parameters, call the function sendChainlinkRequestTo()to make the request. sendChainlinkRequestTo() is an interface method defined in the library provided by Chainlink. Its function is to send data to the Oracle contract along with the transfer.

```
function onGasPriceChange(uint256 gas) public {
  // init SosoDefi.Gas
  SosoDefi.Gas gas = initSosoDefiGas(gas);
  // auto reconcile
  gas.autoReconcile();
  // send to chain
  gas.sendToChain();
}
```

BagAfter the gas fee is obtained, the onGasPriceChange method will be triggered to dynamically adjust the interest rate of the Bag. OnGasPriceChange is defined as follows.

At this point, SosoDifi interest rate automatic adjustment is completed.

OPTION PRICING MODEL

5.1 B-S Model

The B-S Model has the following assumptions:

1.The options are European calls.

2.There are no transaction costs or taxes.

3.The investor can borrow at the risk-free rate.

4.The risk-free rate of interest and the share's volatility is constant over the life of the option.

5.The future share-price volatility can be estimated by observing past share price volatility.

6.The share price follows a random walk and that the possible share prices are based on a normal distribution.

7.No dividends are payable before the option expiry date.

$$\mathbf{C = S \bullet N(d_1) - Le^{-rT}N(d_2)}$$

5.2 B-S Pricing Formula

$$d_1 = \frac{\ln \frac{S}{L} + (r + 0.5 \bullet \sigma^2)\, T}{\sigma \bullet \sqrt{T}}$$

$$d_2 = \frac{\ln \frac{S}{L} + (r - 0.5 \bullet \sigma^2)\, T}{\sigma \bullet \sqrt{T}} = d_1 - \sigma\sqrt{T}$$

where:

• C—The initial reasonable price of the option

• L—option delivery price

• S—the current price of the financial asset being traded

• T—validity period of the option

• r—continuous compounding risk-free interest rate

• σ2—annualized variance

• N()—The cumulative probability distribution function of a normally

distributed variable $\left(\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{d_+} e^{-\frac{x^2}{2}}dx\right)$.

Two points should be stated here:

First, the risk-free interest rate in the model must be continuously compounded. A simple or discontinuous risk-free interest rate (set as R0) is generally compounded once a year, and r requires continuous compounding of interest rates. R0 must be converted to r before it can be substituted into the above formula. The conversion relationship is: r =ln(1+r0)or r0=er-1. For example, R0=0.06, then r=ln(1+0.06)=0.0583. It means that if you use $100 to invest with 5.83% continuous compound interest, you will get $106 in the second year. This result is consistent with the answer calculated directly with r0=0.06. Second, the option validity period T is expressed in relative numbers, that is, the days of the valid period of the option dividing 365. If the option is valid for 100 days, then $T = \frac{100}{365} = 0.274$

YIELD FRAMING

6.1 The Principle of Yield Farming

In view of the important position of liquidity in the SosoDefi protocol, the allocation of SSDF tokens is designed to be proportional to the liquidity on SosoDefi. SSDF tokens will be allocated according to the proportion of the liquidity contributed by each address to the total liquidity of SosoDefi.

SSDF uses a dynamic interest rate model. Every increasing of CDP will increase the reward. The initial contract will reward 20 SSDF per block. With the increase of LP, the reward SSDF per block increases by 20% to 80%. As an example, 24 SSDF～36 SSDF may appear in the future as the reward of a unit block. There is no upper limit for yield farming. The stronger the liquidity, the higher the reward for the CDP account.

The formula of the excitation model of joint yield farming is as

$$V_{(n)} = \sum_0^N \frac{S(n)}{L(n)} * \frac{(sToken+sDao)}{P(n)}$$

follows

•V(n): The number of SSDF produced by joint farming in a single day

•S(n): The pledged sToken mining time period

•L(n): Joint yield farming annualized income

•sToken: the number of sToken held by a single CDP account

•sDao: the joined sDao organization

•P(n): Yield Farming ratio

•S(n), the tokens produced per second in the Ethereum block. Initially, each block generates 20 SSDF rewards.

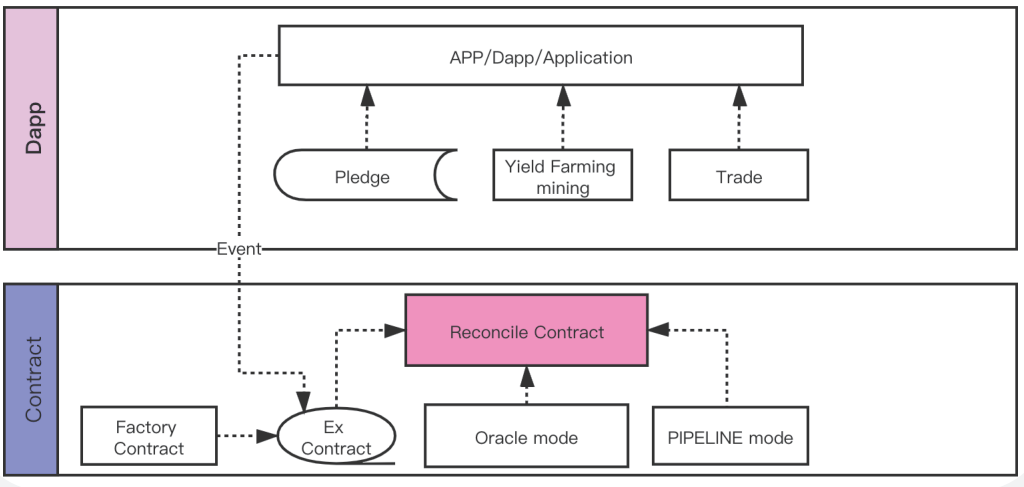
6.2 The Profit Model of Yield Farming

SosoDefi Yield Farming has the following characteristics:

1. Users can participate in the governance

2. High annual rate of rewards

3. Multi-platform pledge



When the user triggers the operation of the SosoDefi platform (staking, providing yield farming, trading), an event will be sent to the SosoDefi contract. The SosoDefi contract structure is as shown in the figure above.

CONCLUSION

The SosoDefi protocol relies on the pool model to provide high liquidity. The loan is backed by collateral and is represented by sTokens (derivative tokens representing accrued interest). Parameters such as interest rate and loan-to-value are token-specific.

SosoDefi has improved the current ecosystem of DEFI and brought the following innovations to the DEFI ecosystem:

- Convenient and safe investment and financing platform;
- Using the oracle model to dynamically adjust the loan rate;
- The PIPELINE model is convenient to simplify the yield farming operation.