veBoosters - maximising boost for Curve LP

Stake DAO research team

July 2023

1 Introduction

For the past couple of years, providing liquidity on Curve was fairly simple. There were two main options: users could provide liquidity directly on Curve boosted by their own veCRV, or they could deposit their LP tokens on Convex and benefit from Convex's veCRV, in exchange for a fee.

Now that multiple solutions are emerging (Stake DAO, Yearn, Conic, etc.), it becomes more and more difficult for users to get the optimal yield on their LP tokens.

Furthermore, this question is valid for several other protocols based on veTokenomics, such as Balancer or Pendle for example, and probably more to come in the future. With veBoosters, Stake DAO aims to provide the most optimal answer to offer the best yields to liquidity providers on Curve, and in the future on other protocols.

2 How does boosting work?

Boosting is a key feature of veTokenomics, introduced by Curve Finance (for more information, please refer to Curve DAO whitepaper). It allows users locking CRV to receive a better incentive yield on the liquidity they provide to the protocol than users who don't have any veCRV.

The formula ruling this boost was introduced by Michael Egorov and reused (or variations of it) by most protocols using veTokenomics. It works with an working balance (b_u^*), which depends on users balance in the liquidity gauge (b_u) and their veCRV balance (w_u).

$$b_u^* = min\left(0.4b_u + 0.6S\frac{w_u}{W}, b_u\right)$$

Where S is the total supply of the liquidity gauge, and W is the total supply of veCRV.

Users' stream of reward are proportionate to these computed working balances. The boost is displayed by Curve is the ratio of the user's working balance over the minimum working balance.

$$Boost_u = \frac{b_u^*}{0.4b_u}$$

To put it in a nutshell: the bigger your share in total veCRV, the bigger your boost (capped to 2.5x), and the bigger your share in the liquidity gauge, the smaller your boost.

3 A need for boost optimised strategies

There are two kind of users on Curve: those who lock CRV and boost their own position, and those who go through boost aggregators such as Convex, Yearn, or Stake DAO. For those who are willing to go through boost aggregators, the main question is "where will I find the best yield?".

Now that we know how veCRV boost works, we can have a broad look at what the average boosts look like for those three boost aggregators as of today (July 2023).

	Convex	Yearn	Stake DAO
Share of Curve's TVL on Eth.	c.85%	c.5%	c.0.85%
Share of veCRV	48.6%	10.4%	7.5%
Theoretical overall boost	1.8x	2.5x	2.5x
Additional TVL while keeping max boost	0	c.$200m	c.$250m

Obviously this is just an approximate since boost needs to be approached on a gauge by gauge basis, but this table highlights the inefficiency of deposits via those different platforms. 85% of deposits are going through Convex while the protocole "only" has 48.6% of veCRV, while Yearn and Stake DAO have a bigger share in total veCRV than in total Curve TVL.

There are several reasons for this, but mainly, it is not only a boost question. Those different protocols have different fees and incentives politics. Therefore, to compare those three protocols, it is important to compute for each gauge an adjusted boost. This is very tricky, and most users don't bother optimising their deposits to maximise their boost.

That's why a contract doing it automatically and adjusting deposits to optimise the real boost for users would bring value to liquidity providers as it would increase their yield and its sustainability.

4 Optimising deposits to maximise user boost

Let's look into how we can maximise user boost by splitting LP deposits through different platforms.

The three platforms offering boost aggregation are Convex, Yearn and Stake DAO. While Convex and Stake DAO have the same structure (depositing users' LP through their locker without compounding rewards), Yearn is already doing a basic optimisation work by depositing through their own locker until maximum boost is not achieved anymore, and then depositing on Convex, and compounding claimed rewards. As we will see later, the yield improvement by splitting deposits is good, but even if perfectly optimised not enough to justify a second layer of fees on top Convex. Therefore, in the perspective of smart contract implementation and gas optimisation, we will focus on optimising deposits through Convex and Stake DAO.

4.1 Assuming no fees and incentives

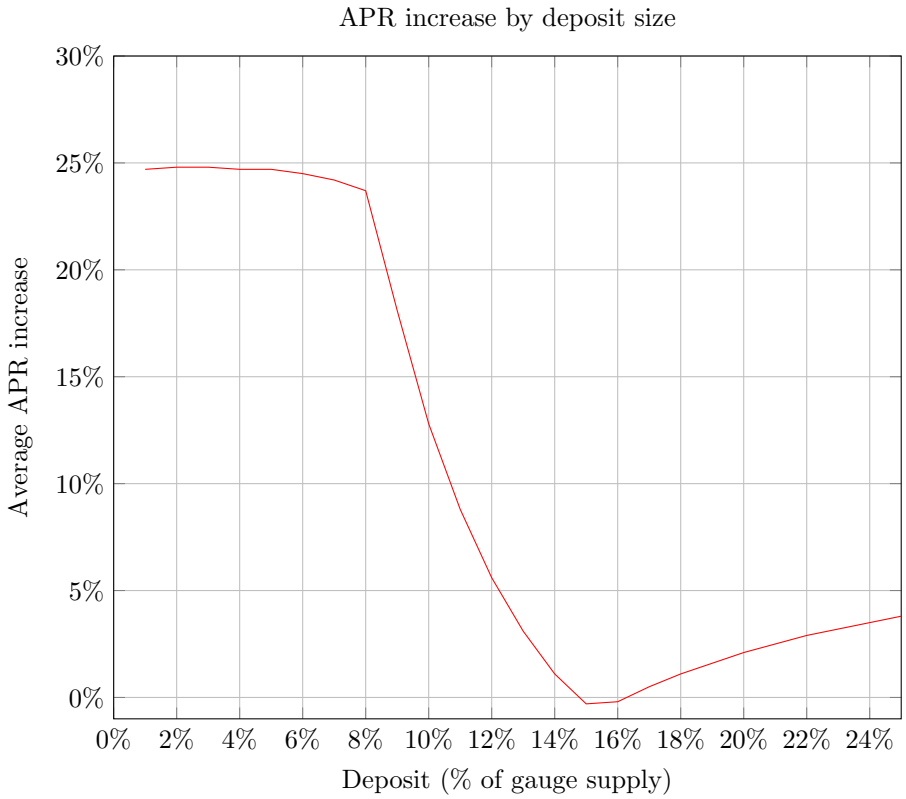
As a first step, let's look at what this problem would give assuming no fees nor incentives. We will look at the target gauge balance of Stake DAO (b_{sd}) that would make Stake DAO's boost (B_{sd}) lower than Convex's boost (B_{cvx}).

$$B_{sd} < B_{cvx} \Leftrightarrow \frac{b^*_{sd}}{0.4b_{sd}} < \frac{b^*_{cvx}}{0.4b_{cvx}}$$

$$B_{sd} < B_{cvx} \Leftrightarrow \frac{0.4b_{sd} + 0.6S\frac{w_{sd}}{W}}{b_{sd}} < \frac{0.4b_{cvx} + 0.6S\frac{w_{cvx}}{W}}{b_{cvx}}$$

$$B_{sd} < B_{cvx} \Leftrightarrow \frac{w_{sd}}{b_{sd}} < \frac{w_{cvx}}{b_{cvx}}$$

As we see, assuming no fees nor incentives, the optimal split between two platforms is when you equalise the ratio $\frac{veCRVshare}{TVLshare}$ for both platforms. This enables us to compute a target balance for Stake DAO b_{sd} which we will try to maintain by depositing through Convex and Stake DAO to optimise user APR. When we test this approach over a large range of over 150 gauges deemed relevant (added to gauge controller, and supply superior to 0), we get mixed results. When additional deposits are fully going to Stake DAO to rebalance boost, the APR increases by c.25% and boost by 0.40x, but when deposits reach a balanced state, the achieved APR is hardly better than when going straight to Convex, and even going a bit into negative territory. That's because when you have similar boost on both sides, the incentives you receive on Convex give you a better APR than on Stake DAO.



We therefore need to take into account fees and incentives to reach a really scalable value-added strategy.

4.2 Adjusting for fees and incentives

Convex and Stake DAO fee structure are fairly similar, with no management or withdrawal fee, and just a 17% and 16% performance fee respectively. Both strategies provide incentives, but Stake DAO incentives are quite volatile and depend on veSDT votes, while Convex incentives are directly proportional to CRV farmed, i.e.

to pools' APR. Due to their nature, it is very difficult to account for SDT incentives in our optimisation, we therefore choose to assume no incentives are being distributed on Stake DAO side.

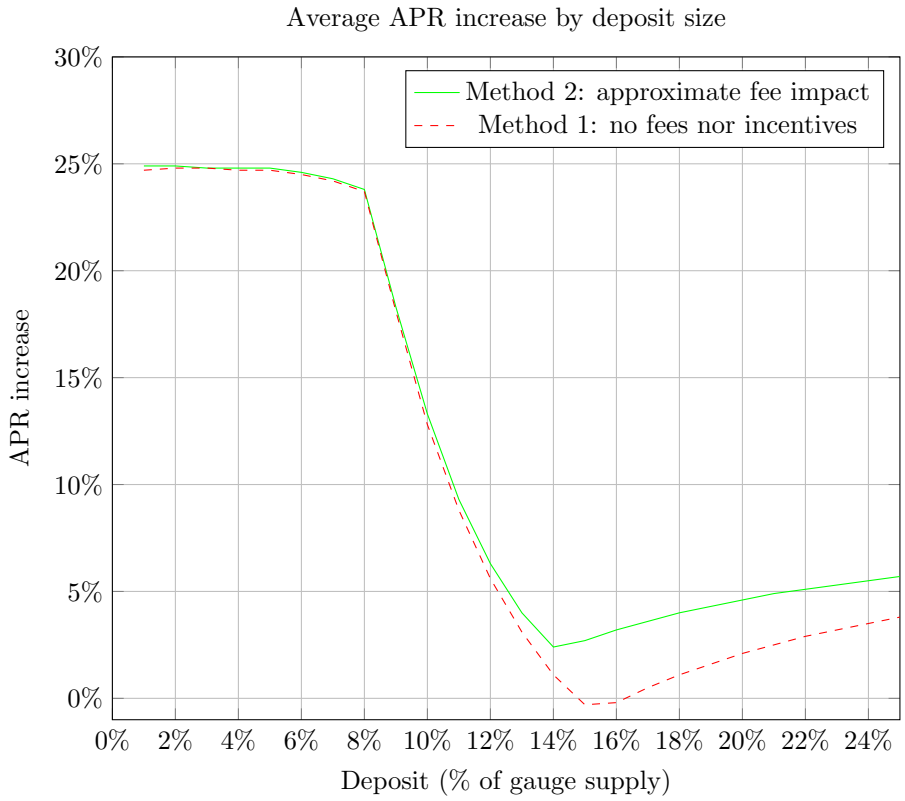
The impact of fees (f) and incentives (i) on APRs and boosts is linear, so the problem we are trying to solve is now:

$$B_{sd}(1 - f_{sd}) < B_{cvx}(1 - f_{cvx} + i_{cvx})$$

Let's introduce the variable $F_{cvx} = f_{cvx} - i_{cvx}$. We will see in section 4.4 how Convex incentives can be calculated in a scalable and on-chain manner. A rough approximate that naturally comes to mind would be to multiply the target balance of Stake DAO by this fee ratio:

$$b_{sd} = b_{cvx} \cdot \frac{w_{sd}}{w_{cvx}} \cdot \frac{1 - F_{cvx}}{1 - f_{sd}}$$

This is just an approximation since the boost formula is not linear in itself and depends on user balances, but it already provides satisfactory results.



As we can see, taking fees and incentives into account even with this approximation is already providing a significant improvement compared to the first simulation. It notably behaves much more efficiently in a balanced state, ensuring that the user will always receive an APR at least 2.4% higher than when going straight to Convex. As deposits keep coming, the average APR gain progressively increases towards a 7% premium compared to depositing directly on Convex.

4.3 Adapting to boost non-linearity

As mentioned previously, Method 2 is just an approximation due to the non-linearity of the boost. Let's see if we can improve our approach while keeping something that

can easily be implemented in a smart contract without excessive gas fees.

Let's use $B^*_{sd} = B_{sd}(1 - f_{sd})$ and $B^*_{cvx} = B_{cvx}(1 - F_{cvx})$.

To find the target Stake DAO balance that needs to be reached to maximise user APR, we need to solve:

$$B^*_{sd} < B^*_{cvx} \qquad (i)$$

We can assume that when this balance is reached for a given gauge, neither Convex nor Stake DAO have maximum boost, which enables us to get rid of the *min* formula.

$$(i) \Leftrightarrow (1 - f_{sd})\frac{0.4b_{sd} + 0.6\frac{w_{sd} \cdot S}{W}}{0.4b_{sd}} < (1 - F_{cvx})\frac{0.4b_{cvx} + 0.6\frac{w_{cvx} \cdot S}{W}}{0.4b_{cvx}}$$

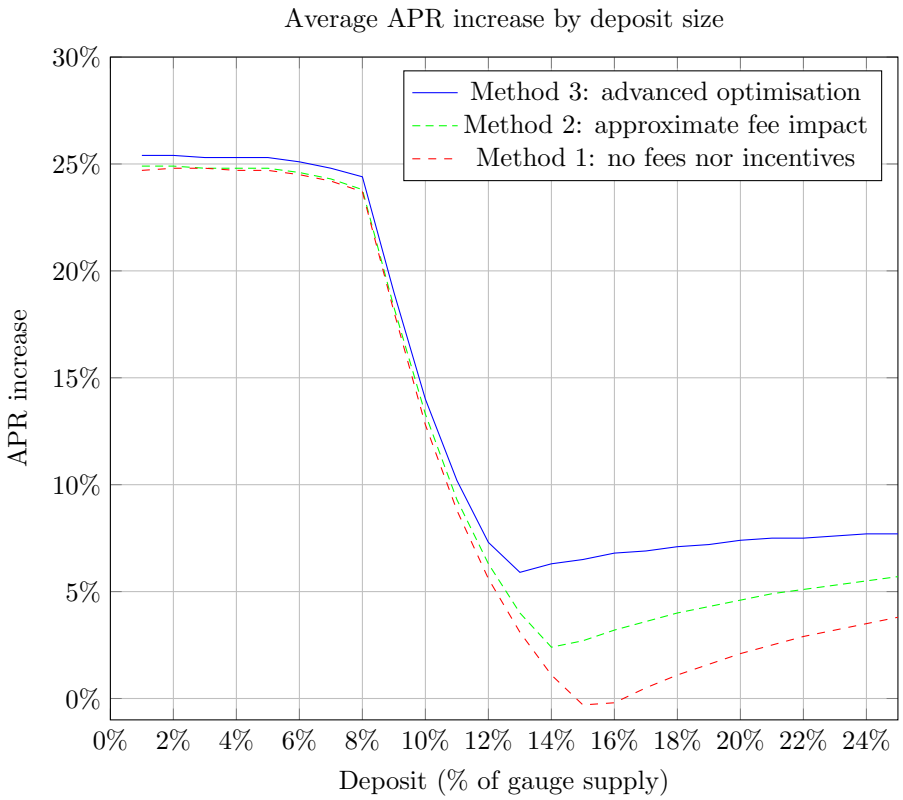
$$(i) \Leftrightarrow \frac{w_{sd}S}{b_{sd}W} < \frac{2(f_{sd} - F_{cvx})}{3(1 - f_{sd})} + \frac{w_{cvx}S}{b_{cvx}W} \cdot \frac{1 - F_{cvx}}{1 - f_{sd}}$$

As $S = S_0 + x$ and $b_{sd} = b_{sd0} + x$ where S_0 and b_{sd0} are initial deposits, and x is the additional deposit needed to reach the target SD balance, we have here a non linear problem which leads to a second degree polynomial problem with complex roots. The complexity of this solution is such that gas costs involved in reaching it are higher than the yield optimisation it provides.

However we can assume that for small deposits, $S \approx S_0$, and get back to a linear problem again. With this assumption, we have:

$$(i) \Leftrightarrow b_{sd} > \frac{3(1 - f_{sd})b_{cvx}w_{sd}S_0}{2(f_{sd} - F_{cvx})b_{cvx}W + 3w_{cvx}S_0(1 - F_{cvx})}$$

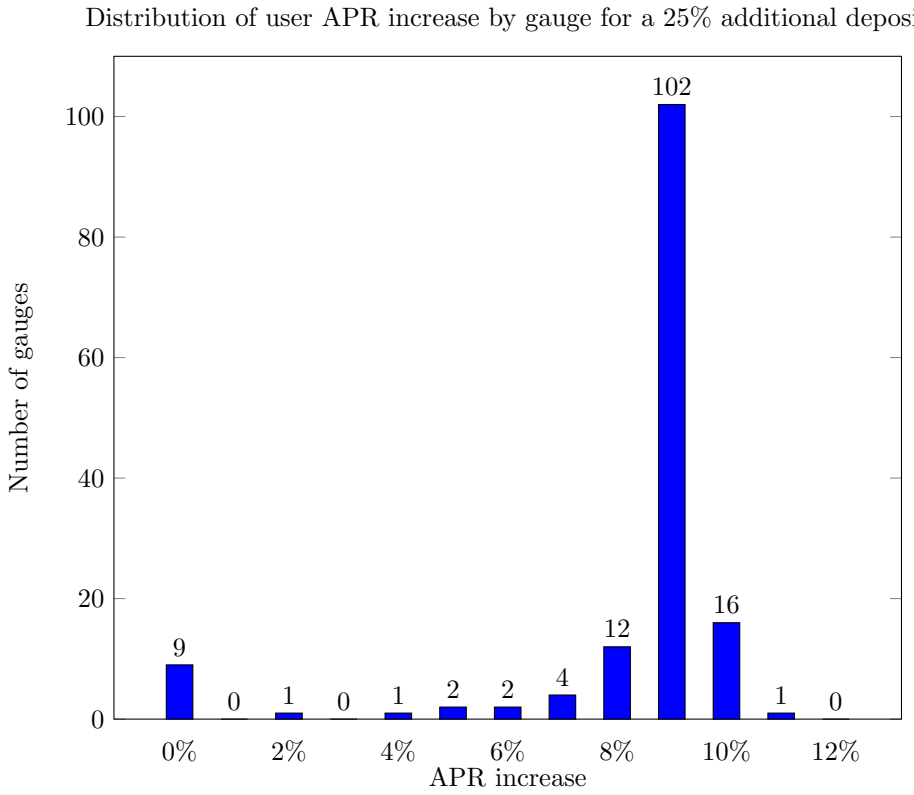
When back testing this solution on the same gauges as before, we obtain another significant improvement compared to the two other solutions.



As we see on the chart, this advanced optimisation allows an increase of the APR of at least 6% compared to depositing straight on Convex, and a sustainable 8% premium compared when deposits start scaling. This corresponds to an average 0.13x boost increase for users.

As we said earlier, it is a very nice increase but doesn't justify an additional performance fee on top of Convex, that's why it's important that this strategy doesn't charge fees for the share of deposits that go through Convex.

Now that we looked at the average impact on a large amount of gauges, the next question is how are those APR increase distributed for those various gauges.

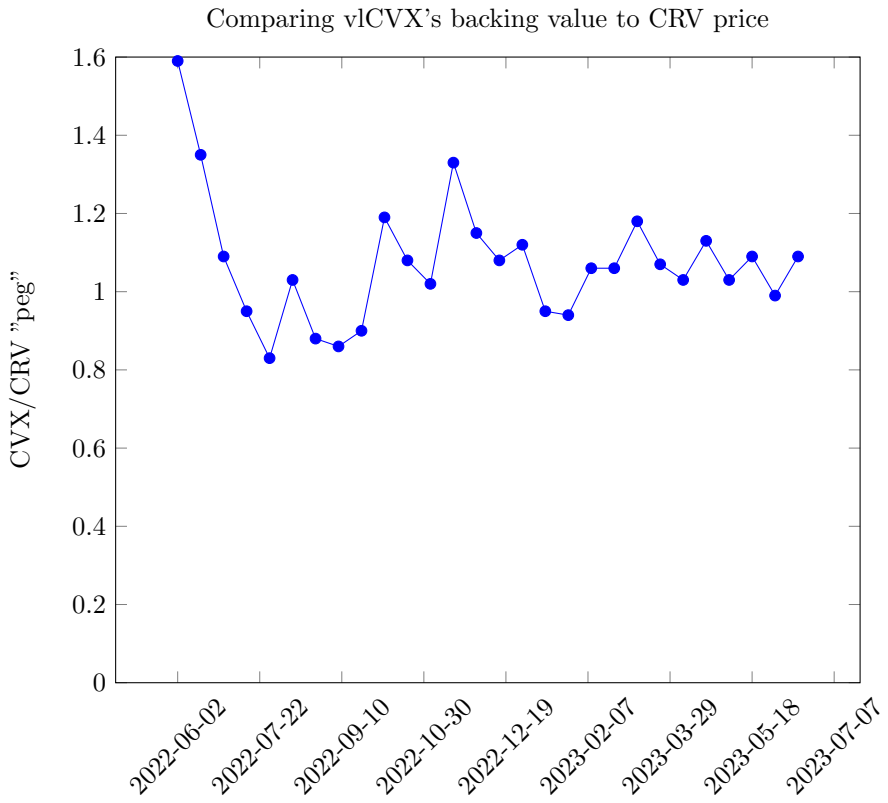


We see that in the long run (vast amount of total deposits in the strategy), the majority of gauges are concentrated around the 8% to 10% range. 9 gauges (6% of the 150 gauges from this sample) have no improvement at all. Those are the ones for which even with a 25% additional deposit, Convex still keeps the maximum boost and there is no point in splitting between protocols. It's a rare scenario and in the long run, should the market be efficient, it shouldn't happen for the largest gauges.

With the results obtained with this implementation, and the additional gas expenses needed to go one step further in terms of precision, we believe this method 3 is the optimal point we can reach. However, we will implement the smart contract in a way that allows for easy modification of the formula in case a better solution is found at a later stage.

4.4 How to calculate CVX incentives?

One of the challenges of optimising deposits between Stake DAO and Convex is how to account for incentives distributed in a different token than CRV, and for which a different price impacts the APR. A solution could be to use an oracle, but it would be volatile and introduce an external dependency. To avoid this, we need to make another assumption. CVX value follows the one of its CRV backing. When we confront this assumption with the market, we see that the price of CVX actually moves as if the locked CRV only backed vlCVX. Assuming the locked CRV are indeed backing vlCVX, here is the "peg" we can compute for CVX price compared to its CRV backing.



We see that for the past year, CVX's price oscillated in a range of +/-20% it's locked backing. This is a comfortable result that we can use to assume that CVX's price is equal to the CRV backing of vlCVX.

That being said, we just need to use the formula of CVX inflation depending on CRV farmed to have i_{cvx}, the incentive boost provided by CVX incentives.

$$i_{cvx} = \left(1 - \frac{CVXsupply}{10^8}\right) \cdot \frac{w_{cvx}}{vlCVXsupply}$$

4.5 How to deal with FXS incentivised pools?

As some of Convex positions on Curve are entitled to receive additional FXS rewards (it's the case for FraxBP pools and frxeth pools), which they can boost with their FXS locker. This leads to Convex Frax pools being strictly superior to Convex Curve pools. However, as for SDT incentives, FXS incentives depend on veFXS votes, and is not easy to integrate in an on-chain optimisation.

In that case, to maximise user APR, we simply replace the normal Convex deposit by a Convex Frax deposit, and we adjust the incentives boost with an arbitrary factor that can be changed easily to reflect the average amount of FXS incentives directed to Convex gauges. Initially this factor will be set at 0.25x to stay conservative and ensure the veBOOSTERS are not over-weighting Stake DAO, and as a consequence offering a lower yield than direct Convex staking.

5 Smart contract implementation

5.1 Architecture

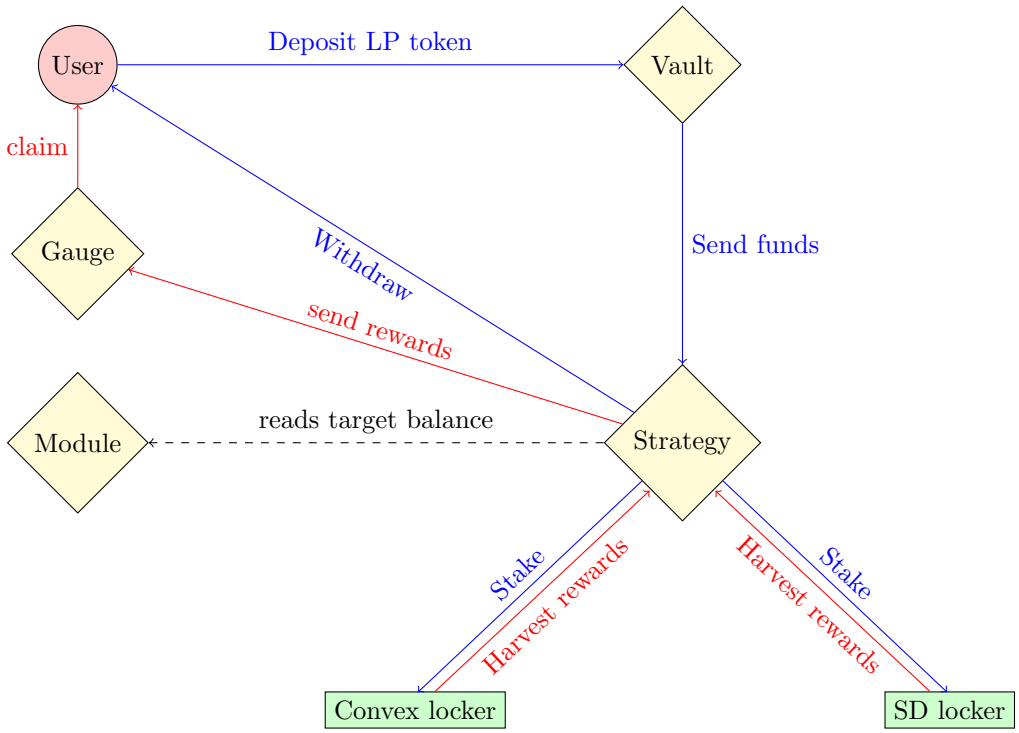
As mentioned earlier, it is important that this optimisation does not lead to additional gas costs for users. For this, we keep the same architecture as the current Stake DAO strategies which provide gas cost mutualisation. Users deposit their LP token in a vault, and when it holds enough deposits, someone can call the *earn* function to move LP tokens from the vault to the strategy, against a 0.1% fee. If a user doesn't want to pay thei gas mutualisation fee, they can simply deposit their LP tokens and call the earn function in the same transaction.

Once funds have been sent to the strategy, it will then allocate them between the Stake DAO locker and the Convex locker depending on whether the target Stake DAO balance has been reached after this deposit. The funds will go to Stake DAO until the target balance is reached, and to Convex after that.

We adopt a modular approach for the computation of Stake DAO's target balance, so that we have flexibility in the future for adapting the formula to potential further improvements. The formula which will be implemented initially is the formula described in subsection 4.3 ("method 3"), with the adjustments described in subsection 4.4.

Withdrawals are performed directly from the strategy, first withdrawing from Convex until strategy balance equals Stake DAO's target balance, and then from Stake DAO. A deposit + withdraw can therefore be used to re-balance the strategy if needed.

A *rebalance* function is implemented to force optimisation in case of big movements in TVL leading to over exposure to one or the other locker.



5.2 Fees

Fees do not change compared to the current version of Stake DAO strategies. Besides the gas mutualisation deposit fee that users can choose to pay or not, there is a performance fee on top of CRV rewards farmed via the Stake DAO locker only. No fees are taken on CRV, CVX, and FXS farmed via the Convex locker, a part from a 1% harvesting fee that go to the user that will harvest the strategy, to ensure its efficiency.

The performance fee on CRV rewards farmed through the Stake DAO locker represents a total of 17% performance fee, split as follows:

- 5% going to veSDT holders
- 2% going to Stake DAO's treasury
- 8% to reward sdCRV depositors
- 1% harvesting fee to ensure the strategy stays efficient

Modifications to the fee structure can be voted by veSDT holders on Stake DAO's governance platform.

5.3 Implementation

The Vault and the Stake DAO locker contracts already exist, written in Solidity, and will not need any migration from users. The liquidity gauge contract already exist

The Strategy contract as well as the Module were implemented in Solidity.