

Farming program

Concepts and comparison to the previous version

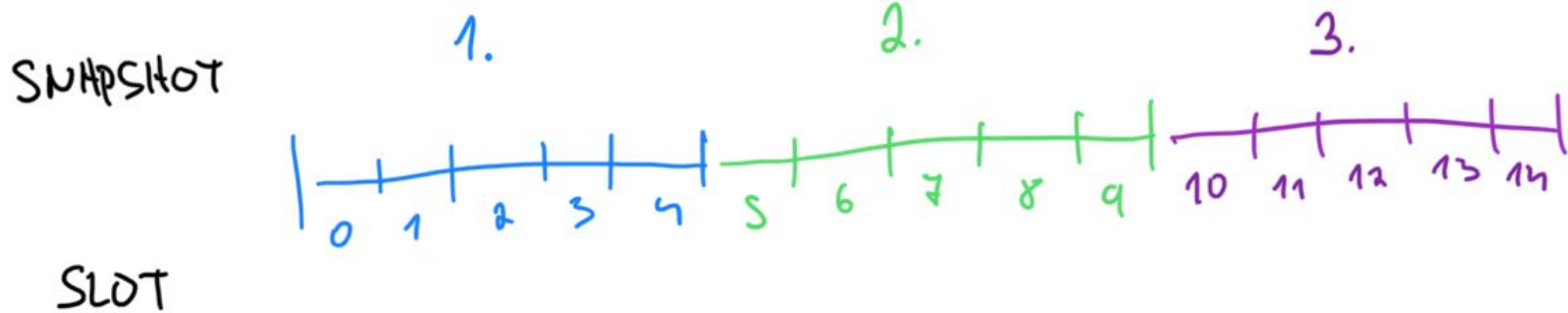
Accounts - Farm

- Iteration on prev version's **FarmingState** + **SnapshotQueue**
- Staking mint
- Staking vault
- Recorded history as snapshots ring buffer
- Harvests
 - Mint
 - Vault
 - Tokens per slot (ρ) : how many tokens to be divided between all farmers per slot
 - Harvest period start slot, end slot



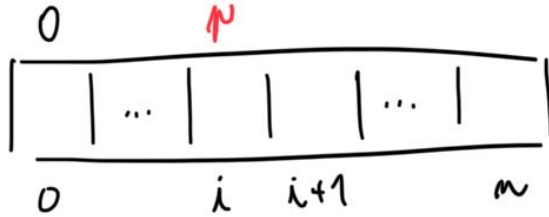
Recorded history

- Snapshots of ~ equal number of slots with constant emission rate and constant staked amount



Recorded history

queue



QUEUE
ENDS

p

latest snapshot

SNAPSHOT
RGE

ARRAY
INDEX



WRITTEN TO
INDEX 0

ring buffer

SNAPSHOT
STARTS-AT (N)
STAKED (N)

Accounts - Farmer

- Iteration on prev version's **FarmingCalc** + **FarmingTicket**
- Staked tokens (F_s) (*number*), vested tokens (*number*) and vested at slot
 - Σ of all farmers' (vested + staked) = farm's staking vault amount
- Harvests
 - Mint
 - Tokens farmed until F_u
- Calculate next harvest from slot (F_u)

Basic Farmer's interaction with Farm

- *start_farming*(**Farm**, **Farmer**, amount)
 - Amount added to vested tokens counter
 - Earning harvest only from next snapshot
- *stop_farming*(**Farm**, **Farmer**, amount)
 - Amount removed from vested + staked tokens counters

Calculate user's harvest

For every reward token ...

```
Initialize_farming_calc(  
    FarmingCalc,  
    FarmingTicket,  
    FarmingState  
)  
  
calculate_farmed(  
    Pool,  
    FarmingState,  
    SnapshotQueue,  
    FarmingCalc,  
    FarmingTicket  
)
```

For all reward tokens at once ...

```
update_eligible_harvest(  
    Farm,  
    Farmer  
)
```

Calculating harvest from history

F_u	slot of farmer's last harvest
F_s	farmer's staked amount
$w(t)$	snapshot at slot t

Handwritten diagram illustrating the calculation of harvest from history:

The formula is:

$$\sum_{j=N(F_u)}^{N-1} \left(\Delta_{j+1} - \max(F_u, \Delta_j) \right) \cdot \frac{F_s}{N_j}$$

Annotations:

- $N-1$ (above the summation symbol)
- $j = N(F_u)$ (below the summation symbol)
- $\Delta_{j+1} - \max(F_u, \Delta_j)$ (inside the summation term)
- Δ_j (below the summation term)
- $\frac{F_s}{N_j}$ (multiplier)
- N_j (below the multiplier)

Interpretations of the components:

- HOW MANY SLOTS OF j IS FARMER ELIGIBLE FOR** (points to the summation range)
- FARMER'S SHARE IN (Δ_j)** (points to the multiplier)
- TOKENS PER SLOT FOR snapshot j** (points to the multiplier)

Continuous harvest

c

current slot

HOW MANY SLOTS OF LATEST
SNAPSHOT IS FARMER ELIGIBLE FOR

FARMER'S SHARE
IN (0:1)

$$(c - \max(F_m, \Delta_n) + 1) \cdot \frac{F_s}{N_n}$$

TOKENS PER SLOT
FOR LATEST
SNAPSHOT

Claim user's harvest

For every reward token ...

```
withdraw_farmed(  
    Pool,  
    FarmingState,  
    FarmingCalc,  
    vault: TokenAccount,  
    wallet: TokenAccount  
)
```

For multiple reward tokens at once ...

```
claim_eligible_harvest(  
    Farmer,  
    remaining accounts: (  
        vault: TokenAccount,  
        wallet: TokenAccount  
    ) []  
)
```

Token emission history

- Changes to p must be kept until no snapshots refer that much back in time
- Limits update frequency
- Each emission rate is always bounded in time
 - *new_harvest_period*(p , **from_slot** = current slot, **length**)

PDAAs

- *Staking vault*: ["stake_vault", farm]
- *Harvest vault*: ["harvest_vault", farm, harvestMint]
- *Vaults signer*: ["signer", farm]
- *Farmer*: ["farmer", farm, authority]
- *Whitelist Compounding*: ["whitelist_compounding", sourceFarm, targetFarm]

Endpoints overview

Admin

create_farm
add_harvest
new_harvest_period
remove_harvest
set_farm_owner
set_min_snapshot_window
whitelist_farm_for_compounding
dewhitelist_farm_for_compounding

User

create_farmer
start_farming*
stop_farming
claim_eligible_harvest
close_farmer

Permission-less

take_snapshot
update_eligible_harvest
compound_across_farms
compound_same_farm

Automation

- Record history
 - *take_snapshot* for each **Farm**
 - In regular intervals
- Recorder history is limited
 - *update_eligible_harvest* for each **Farmer**
 - At least once per history length
- Compounding
 - Frequency depends on business use case
 - *compound_across_farms*
 - *compound_same_farm*