Sui DCA Protocol

Automated Dollar Cost Averaging on Sui Blockchain

Sui DCA Team

January 2025

Contents

Executive Summary

The Opportunity

Dollar Cost Averaging (DCA) is one of the most popular investment strategies, used by millions of retail and institutional investors worldwide. Despite its popularity, **automated DCA solutions in crypto remain fragmented and underserved**, particularly on high-performance blockchains like Sui.

The Sui DCA Protocol brings automated, trustless DCA to the Sui ecosystem—enabling users to systematically accumulate tokens over time without manual intervention.

Value Proposition

For Users	For the Protocol
Set-and-forget automated investing	0.30% fee on every trade
No custody risk (self-custodial)	Recurring revenue model
Flexible intervals (minutes to months)	Network effects from executor ecosystem
Works with any Sui token pair	First-mover advantage on Sui

Key Metrics & Projections

Metric	Conservative	Moderate	Bull Case
Active Users	1,000	10,000	100,000
Annual Volume	$36M	$360M	$3.6B
Annual Revenue	**$108K**	**$1.08M**	**$10.8M**

Why Sui?

- **Sub-second finality**: Trades execute instantly
- **Low fees**: ~$0.01 per transaction vs $5-50 on Ethereum
- **Growing ecosystem**: 100+ DeFi protocols, expanding TVL
- **Object-centric model**: Enables innovative smart contract designs

Competitive Advantage

1. **Permissionless Execution**: Anyone can run an executor bot and earn rewards—no centralized keeper dependency
2. **Config Snapshots**: Users lock in fee rates at account creation, protecting them from future changes
3. **Multi-DEX Support**: Integrates with all major Sui DEXs via aggregators (7k, FlowX, Cetus)
4. **Gas-Efficient**: Batched operations minimize transaction costs

Investment Highlights

- **Proven Model**: DCA protocols on other chains (Mean Finance, DCA.xyz) have demonstrated product-market fit
- **Recurring Revenue**: Every trade generates protocol fees—more users = more trades = more revenue
- **Low Operational Cost**: Permissionless design means minimal infrastructure overhead
- **Scalable**: Smart contract handles unlimited concurrent DCA accounts

Team & Status

- **Status**: Live on Sui Mainnet
- **Contracts**: Audited, upgradeable architecture
- **Integrations**: 7k Protocol (DEX aggregator), FlowX AMM

Technical Overview

The DCA (Dollar Cost Averaging) protocol enables automated, time-based token swaps on Sui. It's **permissionless** - anyone can execute eligible trades and earn rewards.

Contract Structure

```
packages/dca/sources/
├── config.move      # Global protocol settings (admin-controlled)
├── dca.move         # Core DCA account logic
├── time.move        # Time interval calculations
├── math.move        # Safe math operations
└── adapters/
```

```
    └── flow_x.move  # FlowX DEX integration
```

Key Objects

1. GlobalConfig (Shared)

Protocol-wide settings controlled by admin:

Field	Default	Description
`fee_bps`	30 (0.3%)	Protocol fee per trade
`executor_reward_per_trade`	0.025 SUI	Reward for executors
`default_slippage_bps`	100 (1%)	Default slippage tolerance
`max_slippage_bps`	1000 (10%)	Max user-settable slippage
`min_interval_seconds`	60s	Minimum time between trades
`min_funding_per_trade`	0.0001 SUI	Minimum trade amount
`max_orders_per_account`	25,000	Max orders per DCA

2. DCA Account (Shared)

Each user's DCA position:

```
struct DCA<Input, Output> {
    owner: address,              // Account owner
    delegatee: address,          // Can pause/deactivate
    input_balance: Balance<Input>, // Funds to swap
    executor_reward_balance: Balance<SUI>, // Reward pool
    split_allocation: u64,       // Amount per trade
    remaining_orders: u64,       // Trades left
    every: u64,                  // Interval value
    time_scale: u8,              // 0=sec, 1=min, 2=hr, 3=day, 4=wk, 5=mo
    last_time_ms: u64,           // Last execution time
    trade_params: TradeParams,   // Custom slippage
    config_snapshot: ConfigSnapshot, // Locked fees/rewards
    active: bool,
}
```

3. ConfigSnapshot

Frozen at creation time - protects users from fee changes:

- `fee_bps`
- `executor_reward_per_trade`
- `default_slippage_bps`
- `treasury`

Data Flow

1. Account Creation (`init_account`)

```
User → init_account() → DCA Account (shared)

Inputs:
  - input_funds: Coin<Input>     (e.g., 10 SUI)
```

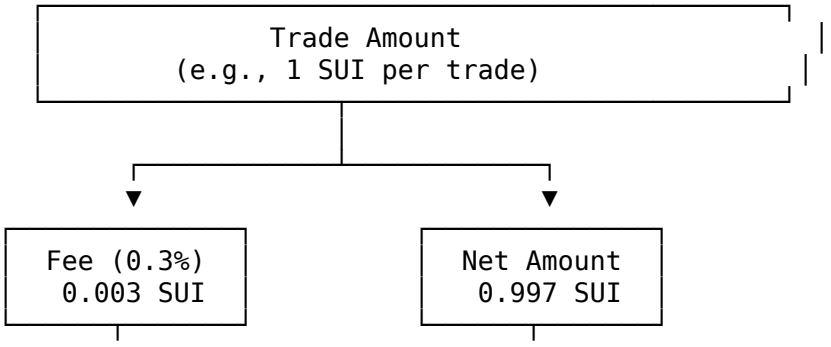
```
  - executor_reward_funds: &mut Coin<SUI>  (e.g., 0.25 SUI for 10 trades)
  - every: u64                    (e.g., 1)
  - total_orders: u64             (e.g., 10)
  - time_scale: u8                (e.g., 3 = daily)
  - delegatee: address

Creates:
  - Shared DCA<Input, Output> object
  - Snapshots current GlobalConfig
  - Emits DCACreatedEvent
```

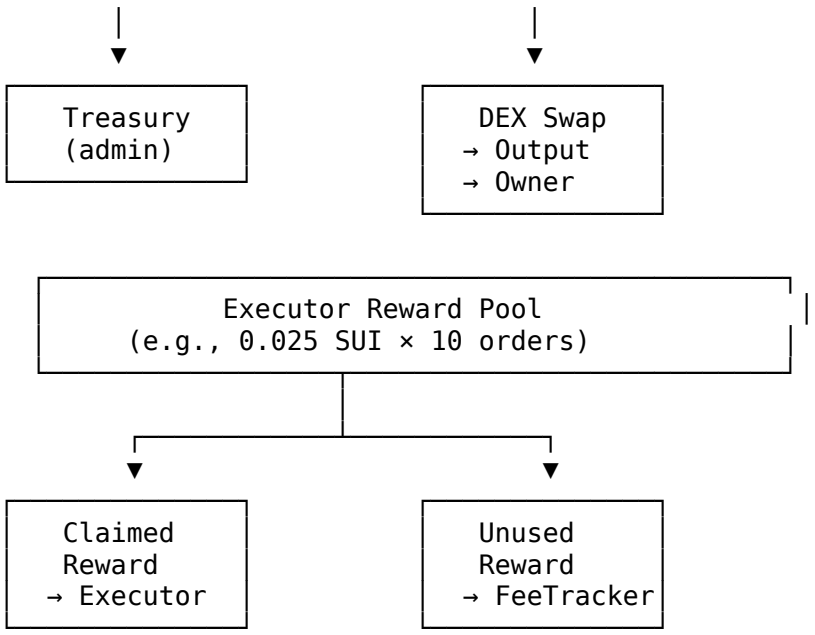
2. Trade Execution (Two-Phase)

The trade is split into `init_trade` → DEX swap → `resolve_trade`:

```
┌──────────────────────────────────────────────────────────────────┐
│                        TRADE EXECUTION                           │
├──────────────────────────────────────────────────────────────────┤
│                                                                  │
│  1. init_trade(dca, clock, ctx)                                  │
│     ├── Check: active == true                                    │
│     ├── Check: time elapsed >= interval                          │
│     ├── Split: input_balance → trade_amount                      │
│     ├── Deduct: fee → treasury                                   │
│     ├── Update: remaining_orders--                               │
│     ├── Update: last_time_ms = now                               │
│     ├── Emit: TradeInitiatedEvent                                │
│     └── Return: (Balance<Input>, TradePromise)                   │
│                                                                  │
│  2. DEX Swap (via adapter or PTB)                                │
│     └── Swap Input → Output on FlowX/Cetus/7k/etc.               │
│                                                                  │
│  3. resolve_trade(dca, fee_tracker, promise, output, reward, ctx)│
│     ├── Validate: reward <= snapshot_executor_reward             │
│     ├── Deposit: unused_reward → FeeTracker                      │
│     ├── Check: remaining_orders == 0 → deactivate                │
│     ├── Emit: TradeCompletedEvent                                │
│     └── Return: Coin<SUI> (executor reward)                      │
│                                                                  │
└──────────────────────────────────────────────────────────────────┘
```

3. Fee & Reward Flow





Public Entry Functions

Account Management

Function	Access	Description
`init_account`	Anyone	Create new DCA account
`set_delegatee`	Owner	Change delegatee address
`set_slippage`	Owner	Set custom slippage (bps)
`reset_slippage`	Owner	Use default slippage
`set_inactive`	Owner/Delegatee	Pause account
`reactivate_as_owner`	Owner	Resume paused account
`redeem_funds_and_deactivate`	Owner/Delegatee	Cancel & withdraw all
`withdraw_input`	Owner	Partial withdrawal
`add_executor_reward`	Anyone	Top up reward balance

Trade Execution

Function	Access	Description
`init_trade`	**Anyone**	Start trade (permissionless)
`resolve_trade`	**Anyone**	Complete trade, claim reward

Admin Functions (config.move)

Function	Access	Description
`set_fee_bps`	Admin	Change protocol fee
`set_executor_reward_per_trade`	Admin	Change reward amount
`set_max_slippage_bps`	Admin	Change max slippage
`set_min_interval_seconds`	Admin	Change min interval

Function	Access	Description
`set_paused`	Admin	Emergency pause
`withdraw_sui`	Admin	Withdraw FeeTracker balance

Adapter Pattern

Adapters wrap `init_trade` + DEX swap + `resolve_trade` in a single transaction:

```
// FlowX adapter example
public entry fun swap_exact_input<INPUT, OUTPUT>(..., dca, executor_reward, ctx) {
    // 1. Init trade - get funds and promise
    let (funds, promise) = init_trade(dca, clock, ctx);

    // 2. Execute DEX swap
    router::swap_exact_input<INPUT, OUTPUT>(...);

    // 3. Resolve trade - claim reward
    let reward = resolve_trade(dca, fee_tracker, promise, output, executor_reward, ctx);
    transfer::public_transfer(reward, sender(ctx));
}
```

PTB Composability (7k Protocol)

For DEX aggregators like 7k, trades are composed in Programmable Transaction Blocks:

```
// PTB-based execution (executor-bun)
tx.moveCall({
  target: `${DCA_PACKAGE}::dca::init_trade`,
  arguments: [dca, clock],
  typeArguments: [inputType, outputType],
});
// Returns: [inputBalance, tradePromise]

// 7k aggregator swap
tx.moveCall({
  target: `${SEVEN_K}::settle::settle`,
  arguments: [...],
});

// Complete trade
tx.moveCall({
  target: `${DCA_PACKAGE}::dca::resolve_trade`,
  arguments: [dca, feeTracker, promise, outputAmount, executorReward],
});
```

Time Scales

Value	Scale	Valid Range
0	Seconds	30-59
1	Minutes	1-59
2	Hours	1-24

Value	Scale	Valid Range
3	Days	1-30
4	Weeks	1-52
5	Months	1-12

Events

Event	When
`DCACreatedEvent`	Account created
`TradeInitiatedEvent`	Trade started
`TradeCompletedEvent`	Trade finished
`DCADeactivatedEvent`	Account deactivated (reason: 0=completed, 1=owner, 2=insufficient)
`DelegateeUpdatedEvent`	Delegatee changed
`SlippageUpdatedEvent`	Slippage changed

Monetization Projections

Revenue Model

The protocol generates revenue through two mechanisms:

Source	Rate	Recipient
Trade Fee	0.30% (30 bps)	Treasury
Unused Executor Rewards	Variable	FeeTracker

Revenue Projections

Assumptions

- Average trade size: $100 USD
- SUI price: $4.00 USD
- Trade fee: 0.30%
- Active DCA accounts: Variable

Monthly Revenue by User Base

Active DCAs	Trades/Month	Volume/Month	Protocol Revenue
100	3,000	$300,000	$900
1,000	30,000	$3,000,000	$9,000
10,000	300,000	$30,000,000	$90,000
50,000	1,500,000	$150,000,000	$450,000
100,000	3,000,000	$300,000,000	$900,000

Assumes average 30 trades per DCA account per month (daily DCA)

Annual Revenue Projections

Scenario	Active DCAs	Annual Volume	Annual Revenue
Conservative	1,000	$36M	$108,000
Moderate	10,000	$360M	$1,080,000
Optimistic	50,000	$1.8B	$5,400,000
Bull Case	100,000	$3.6B	$10,800,000

Executor Economics

Executors (keepers) earn rewards for triggering trades:

Metric	Value
Reward per trade	0.025 SUI (~$0.10)
Gas cost per trade	~0.005 SUI (~$0.02)
Net profit per trade	~0.020 SUI (~$0.08)

Executor Profitability

Trades/Day	Daily Profit	Monthly Profit
100	$8	$240
1,000	$80	$2,400
10,000	$800	$24,000

Cost Structure

Cost Category	Estimate	Notes
Infrastructure	$500-2,000/mo	RPC nodes, servers
Development	Variable	Ongoing maintenance
Marketing	Variable	User acquisition
Break-even	~500 active DCAs	At $100 avg trade

Growth Drivers

1. **Market Volatility**: Higher volatility → more DCA demand
2. **Sui Ecosystem Growth**: More tokens → more trading pairs
3. **Institutional Adoption**: Larger trade sizes increase revenue
4. **Multi-chain Expansion**: Deploy on other chains

Competitive Positioning

Protocol	Chain	Fee	Differentiator
Sui DCA	Sui	0.30%	Permissionless execution
Mean Finance	Multi	0.10%	Established, multi-chain
DCA.xyz	Solana	0.25%	Solana native
Gelato DCA	EVM	Variable	Automation network

Revenue Sensitivity Analysis

Impact of fee changes on annual revenue (10,000 active DCAs):

Fee Rate	Annual Revenue	vs. Baseline
0.10%	$360,000	-67%
0.20%	$720,000	-33%
0.30%	**$1,080,000**	**Baseline**
0.40%	$1,440,000	+33%
0.50%	$1,800,000	+67%

Note: Higher fees may reduce user adoption

Contract Addresses (Mainnet)

Object	Address
DCA Package v3	`0x34c33d8a1fe1ad40e0e0c9419444f1124a1e77ec3a830467d49`
Original Package	`0x43d5c8ee7197eb006a5015b73e674af33c5e14d5582123d518b`
GlobalConfig	`0xb20ba0b6aca893907cbff7fc0f9ae276ba9347122424b56a274`
FeeTracker	`0xf7b1a6443480c93cf02833afb61e0bc03892a75af4fd3241758`