SVP Chain: A Fair Decentralized Perpetual Exchange

Version 1.0

Table of Contents

---

1. Abstract

SVP Chain is a sovereign Layer-1 blockchain purpose-built for decentralized perpetual futures trading. It combines a high-performance on-chain central limit order book (CLOB) with a suite of fairness mechanisms designed to eliminate the structural advantages that harm retail and institutional traders on existing platforms.

The core innovation is a **batch auction matching engine** that replaces continuous price-time priority matching. By collecting orders within each block interval and executing them at a single uniform clearing price, SVP Chain removes the profitability of front-running, sandwich attacks, and latency arbitrage. Two complementary mechanisms — **oracle-based maker protection** and **proposer inclusion monitoring** — address the remaining vectors of adverse selection against liquidity providers and potential manipulation by block proposers.

SVP Chain provides full **EVM compatibility** through a precompiled contract layer, enabling Ethereum developers and users to interact with the fair trading system using familiar tools (Solidity, MetaMask, Hardhat) while ensuring that all transactions — regardless of origin — are subject to the same fairness guarantees.

Building on this composability layer, SVP Chain treats autonomous AI trading agents as **first-class participants** rather than an afterthought. It exposes native programmatic payment (the x402 standard), agent-optimized RPC, an SDK for major agent frameworks, an on-chain agent reputation record, and a protocol-native quant strategy marketplace. Critically, the batch auction design extends the same fairness guarantee to machines: an agent with co-located infrastructure gains no latency advantage over a human trader, because both clear at the same uniform price within each batch. Agents and humans **share the same speed**, competing on the quality of their decisions rather than the speed of their connections.

The batch auction architecture also unlocks a natural **intra-block parallel execution** model. Because each trading pair's auction clears independently, fill execution across non-overlapping markets can proceed concurrently. Combined with infrastructure-level upgrades — including Sparse Merkle Tree (SMT) state storage and asynchronous block execution — SVP Chain's throughput roadmap scales from 1,000 TPS at launch to over 50,000 TPS, delivering institutional-grade performance without sacrificing fairness.

The system is built on Cosmos SDK and CometBFT, inheriting battle-tested consensus, staking, and governance primitives while introducing novel trading-specific fairness and performance guarantees at the protocol level.

---

2. Introduction

Decentralized exchanges have made significant progress in enabling self-custodial trading, but most on-chain CLOB implementations inherit a fundamental problem from traditional finance: **order execution fairness depends on speed**. In a continuous matching model, the first order to arrive captures the best price. This creates a systematic advantage for participants with lower latency — whether through geographic proximity to validators, co-located infrastructure, or privileged access to the transaction mempool.

In traditional finance, this problem manifests as the high-frequency trading (HFT) arms race, where firms spend billions on microsecond advantages. On blockchains, it manifests as Maximal Extractable Value (MEV) — the profit that block producers and specialized actors extract by manipulating transaction ordering within blocks. Research estimates that MEV extraction costs DeFi users billions of dollars annually, functioning as an invisible tax on every trade.

The problem is particularly acute on CLOB-based DEXs because:

- **Order books are transparent.** Pending orders in the mempool are visible before execution, enabling front-running.
- **Block proposers control ordering.** The validator proposing a block decides which transactions to include and in what sequence.
- **Resting orders are vulnerable.** Market makers who provide liquidity through resting limit orders face systematic losses when prices move between blocks — a problem known as adverse selection.

SVP Chain addresses these challenges not through ad-hoc patches, but through a fundamental redesign of the matching engine. By making transaction ordering irrelevant to execution price, the economic incentive for MEV extraction is eliminated at the protocol level.

This redesign arrives as a new class of market participant emerges: autonomous AI trading agents that execute strategies, react to market conditions, and transact without a human in the loop. Most blockchains were designed for human users — interactive signing, rate-limited RPC, and read patterns that assume a person at a keyboard — and treat agents, at best, as a compatibility afterthought. SVP Chain takes the opposite stance: it makes autonomous agents first-class participants while holding them to exactly the same fairness rules as human traders. The batch auction model is what makes this possible without reintroducing the very speed advantage it was built to eliminate, since no participant — human or machine — can convert lower latency into a better execution price. Section 10 describes the agent infrastructure in detail.

---

3. Platform Overview

SVP Chain is a sovereign Layer-1 blockchain built on Cosmos SDK and CometBFT, purpose-built for decentralized perpetual futures trading. This section describes the core capabilities of the platform — the foundation upon which the fairness and performance innovations described in later sections are built.

3.1 Perpetual Futures Trading

SVP Chain supports leveraged perpetual futures contracts — derivative instruments with no expiry date that track the price of an underlying asset. Traders take long or short positions with configurable leverage, posting margin as collateral. Each perpetual market is defined by parameters including tick size, step size, liquidity tier, and maximum open interest.

A **funding rate mechanism** aligns perpetual prices with oracle-reported spot prices. Funding payments are exchanged between long and short holders on an 8-hour cycle, pro-rated by time. When the perpetual trades above the oracle price, longs pay shorts; when below, shorts pay longs. This mechanism ensures long-term price convergence without requiring contract settlement or expiry.

3.2 On-Chain Central Limit Order Book

All order matching occurs on-chain through a fully transparent central limit order book (CLOB). The order book maintains resting bids and asks for each perpetual market, with fills executed deterministically by the validator set during block processing.

Supported order types include:

- **Limit orders** — resting orders that provide liquidity at a specified price
- **Immediate-or-Cancel (IOC)** — fill as much as possible immediately, cancel the remainder
- **Post-Only** — guaranteed to be maker orders; rejected if they would cross the spread
- **Conditional orders** — stop-loss and take-profit triggers that activate when oracle prices reach specified thresholds

3.3 Cross-Margin and Subaccounts

SVP Chain employs a subaccount-based margin system. Each address can operate multiple subaccounts, each with independent positions and collateral balances. Cross-margin subaccounts share collateral across all positions within the subaccount — unrealized gains on one position offset margin requirements on another, improving capital efficiency.

Isolated-margin subaccounts confine risk to a single position, preventing losses in one market from affecting others. Traders choose between capital efficiency (cross-margin) and risk isolation (isolated-margin) on a per-subaccount basis.

3.4 Liquidation and Insurance Fund

When a subaccount's collateral falls below maintenance margin requirements, the position becomes eligible for liquidation. The protocol generates liquidation orders that enter the order book, allowing other market participants to take over the undercollateralized position.

Each perpetual market maintains an isolated insurance fund. When liquidation proceeds are insufficient to cover the deficit — for example, during rapid price movements — the insurance fund absorbs the shortfall. This per-market isolation ensures that a black swan event in one market does not deplete insurance resources for others.

3.5 Oracle Price Feeds

SVP Chain uses a decentralized oracle system where each validator independently sources prices from multiple external exchanges (Binance, Coinbase, OKX, and others). Price reports are attached to consensus votes via CometBFT vote extensions. The on-chain price for each market is the **stake-weighted median** of all validator-reported prices — manipulating it requires controlling more than one-third of total stake, equivalent to compromising BFT consensus itself.

Oracle prices serve multiple critical functions: triggering liquidations, calculating funding rates, and powering the maker protection mechanism described in Section 6.

3.6 Market Listing and Governance

New perpetual markets are added through on-chain governance proposals or through a permissionless listing mechanism that coordinates market creation, oracle configuration, and initial vault liquidity in a single transaction. Governance also controls protocol-wide parameters including fee tiers, funding rate bounds, and margin requirements.

3.7 Fee Tiers and Rewards

Trading fees follow a tiered structure based on trading volume and staking participation. Higher volume or greater stake commitment unlocks lower maker and taker rates. A revenue-sharing system distributes a portion of protocol fees to stakers, affiliates, and liquidity vault depositors, aligning incentives across participant types.

3.8 Liquidity Vaults

SVP Chain provides protocol-managed liquidity vaults where users deposit collateral to provide automated market-making liquidity across perpetual markets. Vault depositors earn a share of trading profits (and bear losses) proportional to their deposit. A share-locking mechanism prevents rapid deposit-and-withdraw cycles that could exploit short-term vault performance.

---

4. Problem Statement

4.1 Latency Advantage

In a price-time priority matching model, orders are matched against the opposite side of the book in the sequence they arrive. Two buy orders at the same price receive different treatment: the earlier arrival fills first, potentially at a better price, while the later arrival faces a thinner book.

This creates a continuous race where faster participants systematically extract value from slower ones. On a blockchain, “faster” means reaching the block proposer’s mempool sooner — an advantage determined by network topology and infrastructure, not by trading skill or market insight.

4.2 Block Proposer Manipulation

The block proposer in a proof-of-stake blockchain has unique power over transaction ordering. In a conventional CLOB architecture, the proposer can:

- **Inject self-signed orders** that other validators have never seen, positioned optimally in the execution sequence
- **Selectively exclude orders** to manipulate the composition of a trading batch
- **Reorder transactions** to place their own trades ahead of large incoming orders

Other validators verify the structural validity of proposed blocks but do not — and cannot, in an asynchronous network — verify that the proposer included every order it received. This gap creates a persistent MEV extraction opportunity for each rotating proposer.

4.3 Maker Adverse Selection

Market makers provide liquidity by placing resting limit orders on both sides of the book. When the external market price moves between blocks, these resting orders become stale — priced at levels that no longer reflect fair value. Arbitrageurs systematically fill these stale orders, capturing the price difference as profit at the market maker's expense.

This adverse selection cost is ultimately passed through to all traders in the form of wider bid-ask spreads. Market makers must price in expected adverse selection losses, reducing the competitiveness of on-chain liquidity relative to centralized venues.

4.4 Existing Approaches and Their Limitations

Several categories of decentralized perpetual exchange architectures have emerged to address portions of the problems above. Each makes meaningful progress but leaves significant gaps that SVP Chain's design targets.

Off-chain orderbook with on-chain settlement. Some application-specific chains move order matching off-chain, with only settlement and state transitions recorded on-chain. This approach achieves low latency and high throughput, but the off-chain matching engine reintroduces centralization risk at the execution layer. Transaction ordering within the off-chain sequencer is opaque — participants must trust that the operator does not reorder, front-run, or selectively delay orders. MEV is not eliminated; it is relocated from validators to the sequencer operator, trading one trust assumption for another.

Application-specific L1 with optimistic execution. Other platforms build dedicated Layer-1 chains optimized for trading performance, achieving sub-second block times and high throughput through aggressive infrastructure optimization. While performance is strong, the underlying matching model remains continuous price-time priority. Faster block times reduce the MEV extraction window but do not eliminate the structural advantage of lower-latency participants. Validators or privileged network participants can still extract value through transaction ordering within each block, and market makers remain exposed to adverse selection between blocks.

MEV-aware transaction sequencing. Some chains implement fair ordering protocols or MEV-aware sequencing rules that attempt to enforce arrival-time ordering or threshold-encrypted mempools. These approaches reduce specific MEV vectors but face fundamental limitations: arrival-time ordering in an asynchronous network is inherently ambiguous (validators disagree on order arrival times), and encrypted mempools add latency and complexity without addressing the core problem that continuous matching rewards speed. The MEV incentive structure remains intact — only the extraction difficulty changes.

AMM-based perpetual protocols. Automated market maker designs (virtual AMMs, concentrated liquidity pools) eliminate the order book entirely, using algorithmic pricing curves instead. This removes front-running of resting orders but introduces different problems: impermanent loss for

liquidity providers, limited order types, poor capital efficiency at scale, and pricing that diverges from centralized markets during high volatility. AMM-based perps have struggled to attract institutional market makers who require the precision and flexibility of limit order books.

What remains unaddressed. Across these approaches, no existing architecture simultaneously provides:

1. Deterministic elimination of intra-block MEV (not just reduction)
2. Protocol-level protection of resting limit orders against adverse selection
3. Transparent, on-chain auditability of block proposer behavior
4. Full EVM composability without fairness gaps between execution paths
5. Order book depth and order type flexibility required by professional market makers

SVP Chain's batch auction architecture addresses this gap — not by optimizing within the continuous matching paradigm, but by replacing it entirely.

---

5. Batch Auction Matching

5.1 Core Mechanism

SVP Chain replaces continuous price-time priority matching with **discrete batch auctions**. During each block interval, incoming orders accumulate in a pending queue without being matched. At block proposal time, all pending orders and eligible resting orders are matched simultaneously at a single **uniform clearing price** per trading pair.

The clearing price is computed to **maximize matched volume** — the price at which the largest quantity of buy and sell orders can be satisfied. All participating buyers pay the same price; all participating sellers receive the same price. The sequence in which orders arrived within the batch has no effect on execution.

5.2 Clearing Price Algorithm

For each trading pair, at the end of the batch window:

Inputs:

- All new buy and sell orders received during the current block interval

- Existing resting orders on the book (after oracle protection filtering; see Section 6)

Procedure:

1. Construct the demand curve: sort all buy orders by price descending, compute cumulative quantity
2. Construct the supply curve: sort all sell orders by price ascending, compute cumulative quantity
3. Find the clearing price P*: the price that maximizes the matchable volume, defined as min(cumulative buy quantity, cumulative sell quantity) at each price level
4. All buy orders with price ≥ P* and all sell orders with price ≤ P* execute at P*
5. If the marginal price level has excess quantity (supply and demand do not exactly balance), allocate fills by:
    - Time priority for resting orders (earlier placement fills first)
    - Pro-rata allocation for new orders within the same batch

```
function computeClearingPrice(bids, asks):
    sort bids by price descending
    sort asks by price ascending

    bidCumulative = cumulativeSum(bids)
    askCumulative = cumulativeSum(asks)

    bestPrice = 0
    bestVolume = 0

    for each priceLevel in union(bid prices, ask prices):
        buyQty  = total bid quantity at prices >= priceLevel
        sellQty = total ask quantity at prices <= priceLevel
        matchable = min(buyQty, sellQty)

        if matchable > bestVolume:
            bestVolume = matchable
            bestPrice  = priceLevel

    fills = allocateFills(bids, asks, bestPrice, bestVolume)
    return (bestPrice, fills)
```

Determinism: The clearing price algorithm is a pure function — identical input order sets produce identical clearing prices and fill allocations regardless of which node executes the computation. All validators independently compute and verify result consistency.

5.3 Order Type Semantics

Order Type	Behavior Under Batch Auctions
Limit	Participates normally; price acts as a ceiling (buy) or floor (sell)
IOC (Immediate-or-Cancel)	Participates in the current batch; unfilled remainder is cancelled
Post-Only	Added to the resting book for the next batch; does not participate as a taker
Conditional (Stop / Take-Profit)	Trigger condition evaluated against the previous batch's clearing price
FOK (Fill-or-Kill)	Checked at clearing time; cancelled if full quantity cannot be matched

5.4 Fairness Properties

Attack Vector	Continuous Matching	Batch Auction
Front-running	Effective: earlier arrival gets better price	Neutralized: uniform clearing price, ordering irrelevant
Sandwich attack	Effective: attacker brackets target order	Neutralized: attacker's orders receive the same price
Latency arbitrage	Effective: faster traders systematically profit	Largely eliminated: no time advantage within a batch
Proposer injection	Effective: proposer can place orders at optimal position	Largely neutralized: injected orders receive the same clearing price

---

6. Oracle-Based Maker Protection

6.1 Core Mechanism

Before each batch auction executes, SVP Chain automatically filters resting orders whose prices deviate significantly from the current oracle price. This prevents market makers' stale orders from being adversely filled after inter-block price movements.

6.2 Protection Band

Each trading pair has a governance-configurable parameter `OracleProtectionBandPpm` (parts per million), defining the maximum allowable deviation of resting order prices from the oracle price.

Rules:

- Resting sell orders priced below `oracle price × (1 − OracleProtectionBandPpm / 1,000,000)` are excluded from the current batch
- Resting buy orders priced above `oracle price × (1 + OracleProtectionBandPpm / 1,000,000)` are excluded from the current batch
- Newly submitted orders are not subject to this filter — active submissions are treated as informed decisions

Example: With `OracleProtectionBandPpm = 5000` (0.5%) and an oracle price of $3,000:

- Resting sell orders below $2,985 are automatically skipped
- Resting buy orders above $3,015 are automatically skipped

Excluded orders are **not cancelled** — they remain on the book and become eligible again once the oracle price moves back into range. This is a per-batch matching exemption, not a permanent removal.

6.3 Parameter Governance

The protection band must balance maker protection against price discovery:

Band Width	Effect	Trade-off
Too narrow (e.g., 0.1%)	Most resting orders excluded	Liquidity collapses, spreads widen, price discovery impaired
Moderate (e.g., 0.3%-0.5%)	Only clearly stale orders excluded	Recommended initial range

Band Width	Effect	Trade-off
Too wide (e.g., 2%+)	Protection rarely triggers	Makers remain exposed to adverse selection

Different trading pairs require different parameters. High-volatility markets need wider bands; stable markets can use narrower ones. Parameters are adjustable through on-chain governance.

6.4 Determinism

The protection mechanism's inputs are exclusively consensus state:

- Oracle price (determined through validator Vote Extensions, identical across all validators)
- Resting orders (on-chain state)
- `OracleProtectionBandPpm` (on-chain parameter)

All validators produce identical filtering decisions, preserving consensus determinism.

6.5 Impact on Arbitrage

A moderate protection band does not eliminate arbitrage — it narrows the arbitrage window. Arbitrageurs can still profit from price discrepancies within the band, ensuring that on-chain prices continue to track external markets. Completely eliminating arbitrage would be counterproductive, as it is the primary mechanism for on-chain price convergence with the broader market.

6.6 Oracle Resilience

The maker protection mechanism introduces a strong dependency on oracle accuracy and availability. If the oracle price is incorrect, the protection band may filter orders that should participate, or in extreme cases, prevent any batch from clearing. SVP Chain treats oracle resilience as a first-class protocol concern and employs three concrete mechanisms to ensure the system degrades gracefully under oracle stress rather than failing catastrophically.

6.6.1 Volatility-Adaptive Protection Bands

A static `OracleProtectionBandPpm` is miscalibrated by definition — a 0.5% band that works during calm markets is too narrow during a volatility spike, and a band wide enough for extreme moves provides insufficient protection during normal conditions.

SVP Chain dynamically adjusts the protection band width based on recent realized volatility:

```
recentVolatility = stddev(oracle price changes over last N blocks)
adaptiveBandPpm  = baseBandPpm × max(1.0, recentVolatility / baselineVolatility)
```

Behavior:

- During calm markets, `recentVolatility ≈ baselineVolatility`, and the band equals the governance-configured base value
- During volatile conditions, the band widens proportionally to observed price movement, preventing legitimate resting orders from being incorrectly filtered
- A governance-configurable `maxBandPpm` caps the upper bound, preventing the band from widening so far that protection becomes meaningless

Breakout detection: If the absolute price change between two consecutive oracle updates exceeds a configurable threshold (e.g., 2%), the system switches from EMA-smoothed band centering to raw latest oracle price. This ensures the protection band tracks the real market during genuine large moves rather than lagging behind a smoothed average.

Market Condition	Band Behavior
Normal (vol < baseline)	Base `OracleProtectionBandPpm` applies
Elevated volatility	Band widens proportionally (up to `maxBandPpm`)
Breakout (single-block move > threshold)	Band centers on raw latest price, width at maximum

All inputs — oracle prices, block history, governance parameters — are consensus state, preserving determinism across validators.

6.6.2 Graceful Degradation Ladder

Rather than a binary switch between “protected” and “unprotected” operation, SVP Chain implements a tiered degradation model based on oracle health:

Oracle Health Score: Each market maintains an on-chain oracle health score computed from three signals:

- **Freshness:** Number of consecutive blocks since the last valid oracle price update
- **Source coverage:** Number of distinct price sources represented in the latest validator reports

- **Inter-validator agreement:** Standard deviation of validator-reported prices (high deviation indicates source disagreement)

Degradation tiers:

Tier	Trigger Condition	Protocol Response
Normal	Oracle updated within last block; ≥3 sources; low inter-validator deviation	Full maker protection with adaptive band
Stale	Oracle missed 1-3 consecutive blocks	Protection band widens by 2× per missed block; maximum leverage reduced by one tier
Degraded	Oracle missed 4-10 blocks, or source count drops below minimum, or inter-validator deviation exceeds threshold	Market enters **maker-only mode**: only post-only orders are accepted; no aggressive crossing permitted. Existing resting orders remain but cannot be filled by new taker orders. This freezes the order book state, protecting makers from stale fills while preserving their ability to cancel or adjust positions.
Halted	Oracle missed >10 consecutive blocks, or oracle health score falls below critical threshold	Market halts new order acceptance. Existing positions remain open and margin is maintained, but no new fills execute. Liquidation processing continues using the last known oracle price with widened safety margins. Market automatically resumes when oracle health recovers to Normal tier.

Per-market isolation: Degradation is evaluated independently for each market. A BTC oracle failure does not affect ETH trading. This prevents a single oracle disruption from cascading across the platform.

Determinism: All health score inputs are derived from on-chain state (oracle update timestamps, vote extension contents, governance parameters). All validators compute identical health scores and degradation tier assignments.

6.6.3 Oracle Price Change Rate Limiting

To prevent both oracle manipulation and feedback-loop amplification, SVP Chain caps the maximum accepted oracle price change per block:

```
maxDelta       = oraclePrice(previous block) × maxOracleChangePpm / 1,000,000
clampedPrice   = clamp(rawMedianPrice, previousPrice − maxDelta, previousPrice +
maxDelta)
acceptedPrice  = clampedPrice
```

Behavior:

- Under normal conditions, actual inter-block price changes are well within the cap, and the rate limiter has no effect
- During extreme moves, the accepted oracle price converges toward the true price over multiple blocks rather than jumping instantaneously
- During manipulation attempts, the attacker must sustain false reports across multiple consecutive blocks to move the accepted price significantly — exponentially increasing the cost and detectability of the attack

Governance parameters:

- `maxOracleChangePpm` : Maximum per-block price change (recommended: 10,000 = 1% for major pairs, 20,000 = 2% for volatile pairs)
- The rate limit applies to the accepted oracle price used for maker protection and liquidations; raw validator reports are still recorded for monitoring and forensic purposes

Interaction with other mechanisms: Rate limiting complements the adaptive band — the band adjusts to volatility while the rate limiter prevents the oracle itself from making implausible jumps. Together, they bound both the protection band's behavior and the oracle input that drives it.

6.6.4 Additional Oracle Safeguards

Beyond the three primary mechanisms, SVP Chain employs several supporting safeguards:

- **Minimum source diversity:** Validator price reports that reference fewer than a configurable minimum number of distinct external sources are excluded from the stake-weighted median. This prevents a single exchange outage from degrading the median quality.
- **Outlier exclusion:** Individual validator reports that deviate more than a configurable threshold from the running median are excluded before the final median is computed, preventing a compromised or malfunctioning validator from disproportionately influencing the price.

- **Minimum retained depth:** Regardless of protection band filtering, at least N best price levels on each side of the book participate in the batch auction. This ensures baseline liquidity is preserved even under oracle stress.

---

7. Proposer Inclusion Monitoring

7.1 Core Mechanism

In an asynchronous network, it is impossible to cryptographically prove which orders a block proposer has received. A proposer can always claim that a missing order simply did not arrive in time. This makes strict inclusion enforcement impractical — it would penalize honest proposers with slower network connections.

Instead, SVP Chain implements a **passive monitoring and transparency system** that makes proposer inclusion behavior auditable on-chain, creating a deterrent against systematic manipulation without risking false punishment.

7.2 Monitoring Architecture

Validator-Side Recording: Each validator maintains a local set of orders it has observed. When evaluating a proposed block, validators compare the block's order set against their own observations and log discrepancies:

```
For each order I have seen but the proposal omits:
    log (block height, proposer address, missing order hash, local first-seen
time)
```

These records are stored locally and do not participate in consensus.

On-Chain Statistical Aggregation: Periodically (e.g., every 1,000 blocks), validators submit summary statistics as on-chain transactions:

```
ProposerInclusionReport {
    reporter:       validator address
    epoch:          reporting period identifier
    proposer:       reported proposer address
    missed_orders:  count of missing orders
```

```
    total_orders:   total orders expected during this proposer's blocks
}
```

7.3 Evaluation Metrics

Inclusion Rate: The proportion of orders included by a proposer relative to orders observed by a supermajority (>2/3) of validators. By counting only orders seen by most validators, network propagation delays are excluded from the metric.

```
inclusion_rate = included orders / orders seen by >2/3 of validators
```

Deviation Score: A proposer's inclusion rate compared against the median inclusion rate of all proposers in the same epoch. Proposers significantly below the median are flagged as anomalous.

7.4 Response Framework

The monitoring system does not impose automatic penalties. It provides transparent on-chain data to support a graduated response:

Level	Trigger	Action
Informational	Inclusion rate below median	Publicly visible on-chain; displayed by indexer/explorer
Governance warning	Sustained deviation across multiple epochs	Community governance discussion
Governance action	Persistent anomaly with sufficient evidence	Governance proposal to reduce proposer weight

7.5 Why Not Automatic Slashing

Automatic slashing in an asynchronous network carries unacceptable false-positive risk:

- Newly joined validators may not yet have a complete view of pending orders
- Network partitions can cause entire validator subsets to temporarily miss orders
- Geographically distant validators inherently experience higher order propagation latency

Delegating judgment to governance rather than protocol code allows context-sensitive evaluation and avoids mechanical false punishment.

7.6 Synergy with Batch Auctions

Under batch auctions, a proposer's ability to profit from selective exclusion is already limited — excluding an order can only shift the clearing price marginally. The monitoring system's value lies in:

- Providing transparency and deterrence against residual manipulation
- Supplying the community with data to assess whether stronger mechanisms are warranted
- Building a long-term dataset for evaluating the effectiveness of fairness improvements

---

8. Performance

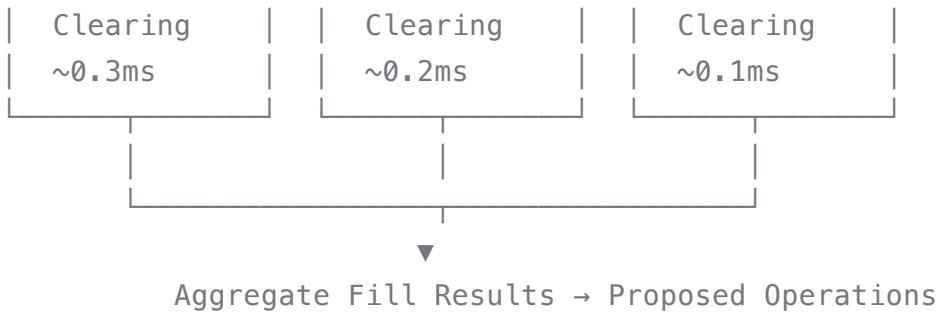
8.1 Throughput Model

SVP Chain's throughput is determined by three stages: order ingestion, batch auction computation, and on-chain state commitment.

Order ingestion: Under the batch auction model, transaction validation only performs signature verification, margin checks, and rate limit enforcement before adding orders to the pending queue. No matching occurs during this phase, making it lighter than continuous matching and increasing per-node order acceptance capacity.

Batch auction computation: The clearing price computation for each trading pair has O(N log N) complexity, where N is the total number of participating orders (resting + new). Sorting is the dominant cost; the clearing price search is a linear scan. For a trading pair with 5,000 orders per block, computation completes in sub-millisecond time.

Multi-market parallelism: Each trading pair's batch auction is independent — pair A's clearing price does not depend on pair B's computation. This enables parallel execution across all trading pairs:



With 100+ trading pairs, total computation time is bounded by the single most active pair, not the sum of all pairs.

State commitment: Fill execution — updating positions, margin, and fees for both parties — is the primary throughput bottleneck. Batch auctions do not increase per-fill state update costs.

8.2 Throughput Targets

SVP Chain's throughput scales through a phased roadmap spanning application-level parallelism and infrastructure-level upgrades. The following table summarizes expected end-to-end transaction throughput at each phase, assuming 1.5-second block times:

Phase	Key Upgrade	TPS (est.)
1	Parallel clearing computation; serial fill execution	1,000
2	Parallel fill execution via dependency graph (see Section 8.5)	4,000-8,000
3	Optimistic concurrent execution with conflict detection	5,000-10,000
4	SMT replacing IAVL; state storage / commitment separation	10,000-20,000
5	Asynchronous block execution (consensus / execution overlap)	20,000-50,000

Phases 1-3 address the application layer. The batch auction model enables per-market parallelism: markets with non-overlapping sub-accounts execute concurrently, with parallel gains increasing as more markets are active (see Section 8.5 for dependency graph analysis).

Phases 4-5 address infrastructure bottlenecks:

- **SMT replacing IAVL (Phase 4):** The IAVL balanced binary tree incurs write amplification and rebalancing overhead under high-throughput state updates. Sparse Merkle Tree (SMT) provides fixed-depth, deterministic key-to-path mapping with no rebalancing, yielding faster writes and more predictable performance. Separating state storage (high-performance engine such as

PebbleDB) from state commitment (SMT for Merkle proofs) further eliminates the constraint of a single tree serving both purposes.

- **Asynchronous execution (Phase 5):** Decouples consensus from execution, allowing the next consensus round to begin while the previous block's state transitions are still being committed. This overlaps computation with I/O, reducing effective block time without additional hardware requirements.

All upgrades are transparent to the application layer — batch auction semantics, maker protection, and inclusion monitoring remain unchanged regardless of the underlying storage or execution engine.

8.3 Latency Characteristics

Metric	Continuous Matching	Batch Auction
Order-to-fill latency	Immediate (within validation phase)	Up to 1 block (~1-2 seconds)
Fill confirmation latency	1 block	1 block
End-to-end latency	1 block	1-2 blocks

The nature of latency changes fundamentally. Under continuous matching, latency *differences* (on the order of milliseconds) are a source of unfairness — faster participants profit at the expense of slower ones. Under batch auctions, all participants wait the same block interval. Latency shifts from a competitive resource to a uniform waiting period. The 1-2 second absolute latency is acceptable for most trading scenarios and is an intentional design constraint for high-frequency strategies.

8.4 Proposal Verification Overhead

Batch auctions require all validators to independently recompute clearing prices during proposal verification. This means the clearing algorithm executes twice per block: once by the proposer, once by each validator.

The clearing algorithm is pure computation (sort + linear scan) with no I/O or network dependency. If it completes in milliseconds during proposal, it completes in milliseconds during verification. Relative to total block processing time (typically in the hundreds of milliseconds), the overhead is

modest. This is the cost of fairness — validators must independently verify clearing results rather than blindly trusting the proposer.

8.5 Intra-Block Parallel Execution

The batch auction structure creates a natural foundation for parallel transaction execution — a significant advantage over traditional sequential processing.

Rationale: Each trading pair produces an independent fill list. Fills update buyer and seller sub-account state (positions, margin, fees) and the pair's order book. When two pairs' fills involve entirely different sub-accounts, no state dependency exists between them, and they can execute concurrently.

Dependency graph construction: Before executing fills, the system builds a conflict graph based on shared sub-accounts:

```
Block fills:
  BTC-USD:  sub-accounts {A, B, C}
  ETH-USD:  sub-accounts {D, E, F}
  SOL-USD:  sub-accounts {A, G, H}
  AVAX-USD: sub-accounts {I, J}

Conflict graph:
  BTC-USD ↔ SOL-USD  (shared sub-account A)
  ETH-USD             (independent)
  AVAX-USD            (independent)

Execution plan:
  Parallel group 1: BTC-USD → SOL-USD  (serialized: shared sub-account A)
  Parallel group 2: ETH-USD            (parallel with group 1)
  Parallel group 3: AVAX-USD           (parallel with groups 1 and 2)
```

Determinism guarantees:

- The dependency graph is derived from the fill list (consensus state), so all validators construct identical execution plans
- Different parallel groups operate on disjoint state keys, ensuring results are equivalent to serial execution
- Within each group, pairs execute in deterministic order (e.g., ascending pair ID), ensuring consistent state update sequencing for shared sub-accounts

- Collateral adequacy checks execute after all related position updates within a group complete, avoiding intermediate-state misjudgments

Expected parallelism:

Scenario	Sub-account Overlap	Expected Parallelism
Retail-dominated: most users trade 1-2 markets	Low	High (approaching total market count)
Institutional-dominated: few large accounts across many markets	High	Moderate (constrained by connected market groups)
Worst case: one sub-account across all markets	100%	None (degrades to serial)

In typical mixed user profiles, most markets belong to independent connected components. A few highly active market makers may link several major markets, but long-tail markets generally remain independent. Expected overall speedup is 3-10x.

Phased implementation:

- **Phase 1 (conservative):** Parallel clearing price computation across markets during block proposal and verification. Fill execution remains serial. This covers the compute-intensive stages without concurrent state writes.
- **Phase 2 (parallel execution):** Introduce dependency graph construction and parallel fill execution, requiring storage-layer support for concurrent writes to disjoint key sets.
- **Phase 3 (optimistic execution):** Assume no conflicts and execute all pairs in parallel, detect write conflicts post-hoc, and re-execute conflicting groups serially. More efficient than pre-computed dependency graphs when conflict rates are low.

8.6 EVM Execution Overhead

Precompile call overhead: EVM transactions invoking CLOB functionality incur a context switch between EVM and Cosmos module execution. Per-call overhead is on the order of microseconds — negligible relative to the cost of order validation and state updates.

Smart contract execution: Complex Solidity contracts (e.g., automated strategy contracts) require EVM bytecode execution before invoking precompiles. The gas mechanism caps per-transaction computation, preventing contract execution from consuming excessive block processing time.

Impact by EVM transaction share:

EVM Transaction Share	Additional Block Processing Time
0% (pure native)	Baseline
30%	+5-10%
70%	+15-25%
100% (pure EVM)	+20-35%

Since precompile invocations (rather than complex contract computation) are the dominant EVM usage pattern, actual overhead trends toward the lower end of these estimates.

8.7 Memory and Storage

- **Pending order queue:** Each order occupies ~200-500 bytes; 10,000 pending orders require approximately 2-5 MB.
- **Inclusion monitoring:** Validators' local observed-order sets use bounded-window hash sets or Bloom filters, with per-block memory overhead in the low MB range. On-chain inclusion reports are summary statistics of a few hundred bytes each.
- **EVM state:** Smart contract deployment and storage use a separate EVM state tree. Storage growth is constrained by gas costs — storage operations are among the most expensive EVM operations, and economic incentives naturally suppress state bloat.

8.8 Additional Scalability Paths

Beyond the phased throughput roadmap (Section 8.2), additional architectural paths can further increase capacity:

- **Vertical scaling:** Shorter block times (lower latency) and larger per-block transaction capacity (higher throughput), bounded by validator hardware and network bandwidth.
- **Market sharding:** Assign different trading pairs to independent batch auction channels, potentially processed by different validator subsets. Since batch auctions are naturally independent per pair, sharding does not affect clearing algorithm correctness. Cross-market atomic operations (e.g., spread trades) require additional cross-channel coordination.
- **Off-chain order collection:** Move order collection to an off-chain relay network. Relay nodes collect and forward orders; validators pull order batches from the relay network at block time.

This reduces validators' mempool management burden but introduces dependency on relay network availability.

---

9. EVM Compatibility

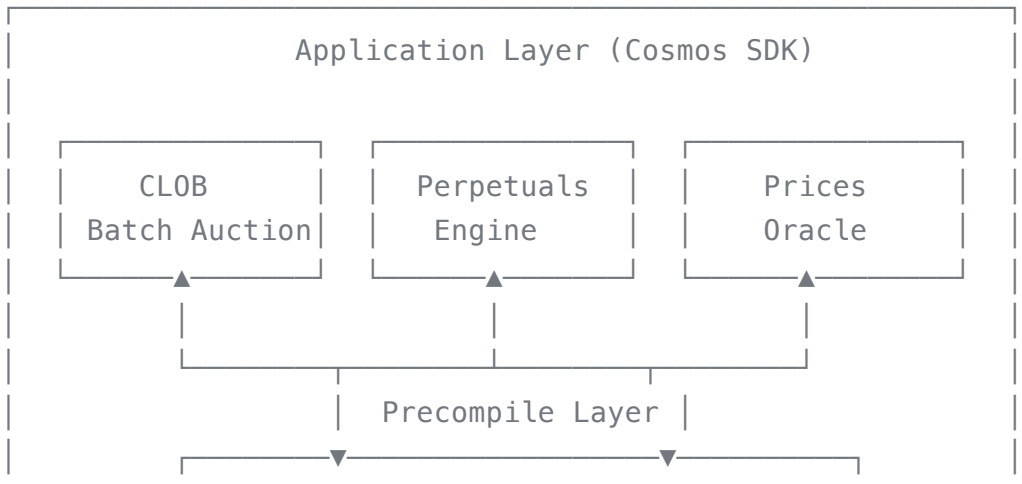
9.1 Design Goals

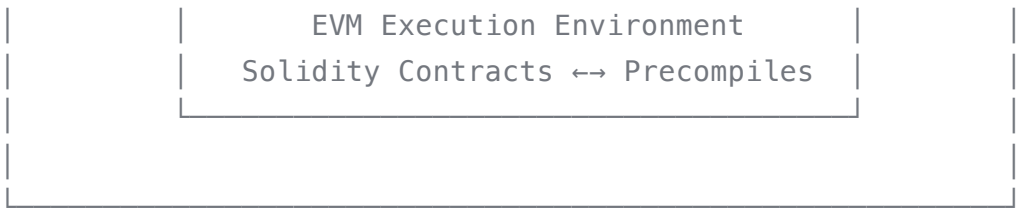
The Ethereum ecosystem hosts the largest smart contract developer community, the most mature toolchain (Solidity, Hardhat, Foundry, ethers.js), and the broadest wallet support (MetaMask, WalletConnect, etc.). SVP Chain's EVM compatibility layer pursues three objectives:

1. **Accessibility:** Traders and developers interact with SVP Chain using standard Ethereum JSON-RPC interfaces and EVM transaction formats — no new SDKs or transaction schemas required
2. **Composability:** Solidity smart contracts can atomically interact with the CLOB trading system, enabling developers to build structured products, automated strategies, and cross-protocol compositions
3. **Fairness parity:** Transactions originating from the EVM layer are subject to the same fairness rules as native transactions — batch auctions, maker protection, and inclusion monitoring apply uniformly, regardless of transaction origin

9.2 Dual Execution Architecture

SVP Chain embeds an EVM execution environment within the Cosmos SDK application layer, forming a dual-path architecture:





Native path: Traders submit Cosmos-format order messages that enter the CLOB module's pending queue directly.

EVM path: Traders submit Ethereum-format transactions. The EVM environment processes the transaction, and smart contracts invoke CLOB module functions through precompiled contracts. Orders enter the same pending queue and are processed identically in the batch auction.

Both paths converge into the same matching pipeline, ensuring fairness mechanisms apply uniformly.

9.3 Precompiled Contracts

Precompiled contracts are system-level contracts deployed at fixed addresses that expose native module functionality as Solidity-callable interfaces. Unlike cross-chain messaging or relay bridges, precompiles provide **synchronous atomic invocation** — contract calls and module state changes complete within a single transaction with no cross-chain confirmation delay.

Core Precompiles:

Address	Name	Functionality
`0x0900`	CLOB	Order placement, cancellation, leverage adjustment
`0x0901`	Sending	Sub-account transfers, withdrawals
`0x0902`	Vault	Vault deposit and withdrawal
`0x0903`	AccountPlus	Account management
`0x0904`	Listing	Market listing operations
`0x0905`	Affiliates	Referral management
`0x0906`	Subaccounts	Sub-account queries and management
`0x0907`	Perpetuals	Perpetual contract parameter queries

Address	Name	Functionality
`0x0908`	Prices	Oracle price queries
`0x0909`	FeeTiers	Fee tier queries

Solidity Interface Example (CLOB Precompile):

```solidity
interface ICLOB {
    /// @notice Submit a limit order to enter the next batch auction
    function placeOrder(
        uint32 clobPairId,
        uint64 quantums,
        uint64 subticks,
        bool isBuy
    ) external returns (bytes32 orderId);

    /// @notice Cancel a resting order that has not yet been filled
    function cancelOrder(bytes32 orderId) external;

    /// @notice Query the estimated clearing price for the current batch (read-only)
    function estimateClearingPrice(
        uint32 clobPairId
    ) external view returns (uint64 subticks);
}
```

9.4 Unified Account Model

SVP Chain employs a unified account model where a single private key maps to both a Cosmos address (Bech32 format) and an EVM address (hexadecimal format), derived from the same `ethsecp256k1` key pair:

```
Private Key → Public Key → Cosmos Address (svp1abc...xyz)
                                └──→ EVM Address (0xABC...XYZ)
```

This means:

- MetaMask users can trade on SVP Chain directly with their existing Ethereum keys

- Positions and margin are shared across native and EVM transactions within the same sub-account
- No cross-chain bridging or address mapping is required

9.5 Fairness Parity

The EVM layer's core design principle is to **create no fairness gap**:

- **Batch auction consistency:** Orders submitted via EVM precompiles enter the same pending queue as native orders. The batch auction algorithm does not distinguish by origin — all orders receive the uniform clearing price.
- **Maker protection consistency:** Resting orders placed via EVM are subject to the same `OracleProtectionBandPpm` filtering. The precompile invokes the same module logic as native messages; the protection cannot be bypassed.
- **Inclusion monitoring consistency:** EVM transactions are tracked in validators' observed-order sets identically to native transactions. Proposer exclusion of EVM-originated orders is captured by the monitoring system.
- **No bypass path exists:** Smart contracts cannot modify CLOB module state except through precompiles, which encapsulate the full validation and queuing logic.

9.6 Smart Contract Composability

The EVM layer's value extends beyond wallet and toolchain compatibility — it makes the fair trading system composable:

- **Automated strategies:** Developers can deploy Solidity contracts implementing grid trading, TWAP execution, conditional triggers, and other algorithmic strategies that participate in batch auctions via the CLOB precompile.
- **Structured products:** Contracts can atomically combine orders across multiple trading pairs to construct spread trades, cross-market hedges, and other multi-leg structures. Because precompile invocations are synchronous, the composite operation either enters the same batch entirely or not at all.
- **DeFi protocol integration:** Lending protocols can read oracle prices via the Prices precompile, trigger liquidations via the CLOB precompile, and manage sub-accounts via the Subaccounts precompile — all within a single EVM transaction.
- **Third-party frontends:** Any frontend supporting EVM JSON-RPC can connect to SVP Chain directly, reducing integration costs for aggregators and third-party interfaces.

9.7 Gas and Fee Model

EVM transactions use Ethereum-standard gas metering with EIP-1559-style dynamic base fees:

- **Gas fees** cover EVM execution costs (contract calls, state storage, etc.) and accrue to validators. Base fees adjust dynamically based on block utilization.
- **Trading fees** are charged separately when orders placed via the CLOB precompile result in fills, using the same Maker/Taker fee schedule as native orders.

The two fee layers are independent, ensuring EVM traders are not disadvantaged by additional gas overhead relative to native traders while the gas mechanism effectively prevents resource abuse on the EVM layer.

9.8 Security Boundaries

- **Permission control:** Precompiles expose only validated operation interfaces. The CLOB precompile's order placement function enforces the same signature verification, margin checks, and rate limits as native messages. Smart contracts cannot obtain elevated privileges through precompiles.
- **Reentrancy protection:** Precompile calls involve a context switch from EVM to Cosmos module execution. Module-to-EVM callbacks are prohibited by design — precompile invocations are unidirectional, eliminating reentrancy vectors.
- **State atomicity:** EVM state (contract storage, account balances) and Cosmos module state are committed atomically within the same block. No inconsistent intermediate state is observable.

---

10. AI Agent Infrastructure

A growing share of trading activity originates not from humans clicking an interface, but from autonomous software agents that interpret market conditions, decide, and transact without a person in the loop. SVP Chain treats these agents as first-class participants. This section describes the infrastructure that supports them and, equally important, why the chain's fairness guarantees apply to machines exactly as they apply to people.

10.1 Design Rationale

Most blockchains were architected around assumptions that hold for a human at a keyboard but break down for an autonomous agent operating continuously across many markets. The result is

that agents are typically supported as a compatibility afterthought rather than designed for. SVP Chain inverts this assumption.

Common chain design (human-oriented)	Requirement for autonomous agents
Interactive, per-transaction signing	Unattended signing under pre-authorized, scoped permissions
Rate-limited public RPC tuned for occasional reads	Sustained high-frequency reads and writes with predictable limits
Point-in-time queries, polled one at a time	Streaming updates and batched queries to track many markets at once
Latency rewarded (faster submission → better fill)	Latency neutralized, so agents are not forced into an infrastructure arms race
Payments mediated by a human approving each transfer	Programmatic, inline payment for data, execution, and services

The remainder of this section addresses each of these requirements in turn.

10.2 Fairness Parity for Autonomous Agents

The most important property SVP Chain offers agents is not a feature but a guarantee: agents are subject to exactly the same fairness rules as human traders, and the batch auction (Section 5) ensures that neither can convert lower latency into a better price.

Under a continuous price-time priority model, an agent with co-located infrastructure and direct mempool access would systematically dominate slower participants — precisely the dynamic that drives the high-frequency arms race in traditional markets. Under batch auctions, that advantage disappears: all orders entering a batch clear at the same uniform price, so the order in which they arrive is irrelevant.

Agent capability	Continuous matching	Batch auction (SVP Chain)
Low-latency order submission	Systematically captures better prices	No advantage: uniform clearing price within the batch

Agent capability	Continuous matching	Batch auction (SVP Chain)
Co-location / privileged mempool access	Enables front-running and sandwiching of slower orders	Neutralized: intra-batch ordering does not affect execution
High-frequency requoting to pick off stale orders	Profits at the expense of resting liquidity	Stale resting orders are filtered by oracle maker protection (Section 6), independent of who submits

The practical consequence is that agents and humans **share the same speed**. An agent's edge must come from the quality of its decisions — its models, signals, and risk management — not from the speed of its connection. Maker protection and proposer inclusion monitoring (Section 7) apply to agent-originated orders identically to any other order; the agent infrastructure described below is a set of access and convenience layers on top of this guarantee, never a bypass of it.

10.3 Agent Account and Authorization Model

Autonomous operation requires an agent to sign transactions without a human approving each one — but delegating the master private key to an always-on process is unacceptable from a security standpoint. SVP Chain resolves this with scoped authorization keys built on the unified account model (Section 9.4).

A master account can authorize one or more subordinate agent keys, each constrained by:

- **Scope** — which precompiles and markets the key may act on (e.g., place and cancel orders on a specific set of perpetual markets, but not withdraw funds)
- **Limits** — maximum notional, position size, or spending over a defined window
- **Expiry** — an on-chain validity window after which the key is automatically inert

The agent holds only the scoped key, never the master key, and the authorization is revocable on-chain at any time. This lets an operator delegate trading authority to an agent without surrendering custody of funds or exposing the account to unbounded risk if the agent is compromised. Because the constraints are enforced by the protocol rather than by the agent's own code, a malfunctioning or malicious agent cannot exceed its mandate.

10.4 x402 Native Payment Integration

Agents frequently need to pay for the resources they consume — data feeds, execution services, or strategy subscriptions — as part of normal operation. SVP Chain natively supports the **x402**

payment standard, which reactivates the HTTP `402 Payment Required` status code so that a client can settle payment for a resource inline, within the same programmatic request flow.

Integrated with the authorization model (Section 10.3), x402 lets an agent discover a price, authorize payment within its scoped limits, and settle on-chain without a human approving each transfer. Payments settle atomically through the same module path as other transfers, inheriting the chain's determinism and finality. This makes machine-to-machine commerce — an agent paying another service, or a user's agent paying a strategy provider — a native protocol capability rather than an off-chain arrangement.

10.5 Agent-Optimized RPC

An agent tracking many markets at high frequency imposes a read/write pattern that generic, rate-limited RPC serves poorly. SVP Chain provides an RPC profile tuned for this workload:

- **Batch queries** — many reads resolved in a single round trip, reducing per-request overhead
- **Streaming subscriptions** — order book, fill, and oracle updates pushed as they occur rather than polled
- **State-diff push** — only the keys that changed since the last block are delivered, rather than re-fetching full state each interval

Together these reduce the read amplification that otherwise makes high-frequency agent operation expensive and brittle, while keeping per-client load predictable for node operators.

10.6 Agent SDK

To minimize integration cost, SVP Chain ships an SDK that exposes its actions — placing and cancelling orders, querying estimated clearing prices, and managing subaccounts — as typed tools consumable by mainstream agent frameworks, including LangChain, OpenAI function calling, and Claude tool use. The SDK wraps both the EVM precompiles (Section 9.3) and native messages, so an agent built on any of these frameworks can act on-chain through familiar abstractions without writing a bespoke integration against the raw transaction format.

10.7 On-Chain Agent Reputation

Because every fill is recorded on-chain, an agent's track record can be derived deterministically from public state rather than self-reported. The protocol surfaces queryable, tamper-evident metrics for each registered agent or strategy, including:

- **Fill / success rate** — the proportion of submitted orders that execute as intended

- **Realized slippage** — average execution price relative to the batch clearing price
- **Failure / liquidation rate** — frequency of liquidations or risk-limit breaches

Because the inputs are on-chain fills, the same metric computed independently by any party yields the same result. This lets a user evaluate an agent before delegating capital to it, and lets a strategy provider's claimed performance be verified rather than trusted.

10.8 AI Quant Strategy Market

Quantitative trading has historically required proprietary infrastructure and was accessible only to specialized firms. SVP Chain generalizes the agent infrastructure above into a protocol-native product: a marketplace where strategy providers publish trading agents and users allocate capital to them.

The flow is enforced end to end on-chain:

1. A provider registers a strategy agent, whose historical performance is visible through the reputation record (Section 10.7).
2. A user allocates capital to a chosen strategy through the authorization model (Section 10.3), into a dedicated subaccount (Section 3.3) — without surrendering custody.
3. The agent trades within its scoped permissions; the subaccount bounds the user's exposure to exactly the capital allocated.
4. Realized profits are split between user and provider according to an on-chain, protocol-enforced performance-fee schedule.

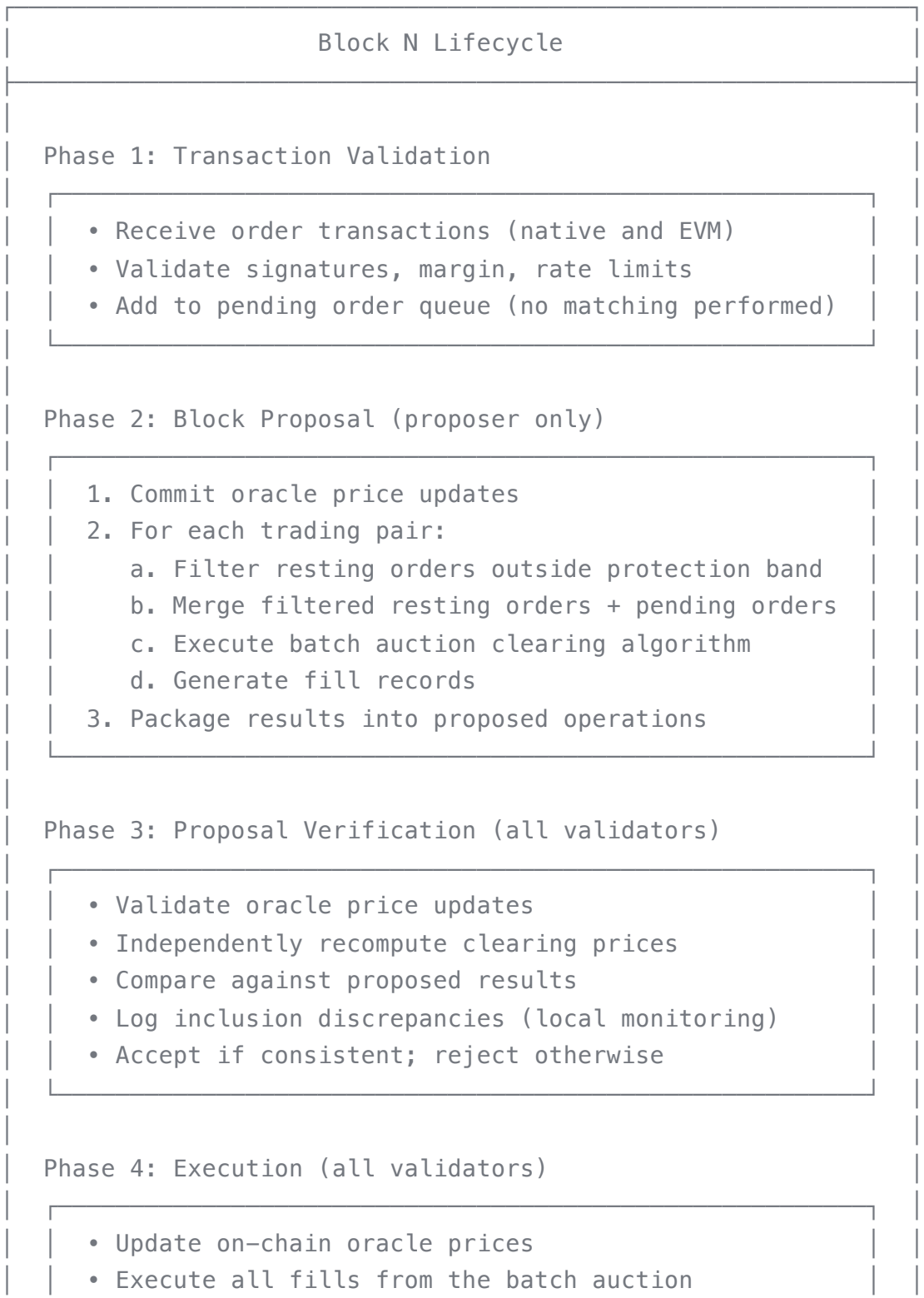
Because allocation, execution, and settlement all occur on-chain, the performance-fee split cannot be circumvented by either party, and results continuously update the provider's reputation record. The effect is to make quantitative strategies — previously the domain of well-resourced firms — available to any user as a composable, verifiable on-chain product.

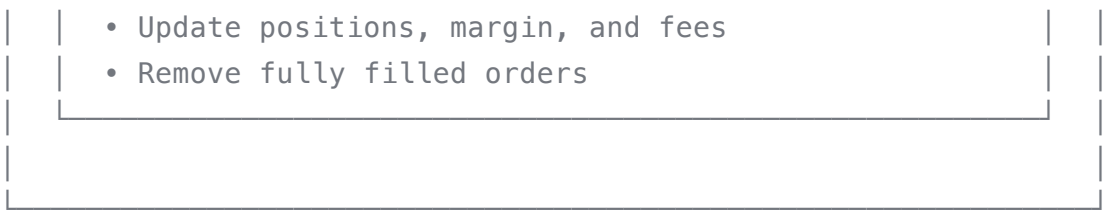
10.9 Future Direction: Inter-Agent Settlement

As agent ecosystems mature, a single trading objective is often decomposed across cooperating agents — for example, a signal agent that decides what to trade, an execution agent that works the order into successive batches, and a risk-control agent that enforces position and drawdown limits. A future extension would provide native on-chain settlement and profit attribution among such agents, so that value generated by a multi-agent workflow can be divided among its contributors under enforceable, transparent terms. This direction is exploratory and is not part of the initial protocol; it is included here to indicate the trajectory the agent infrastructure is designed to support.

11. System Architecture

10.1 Block Lifecycle





10.2 Key Architectural Properties

Property	Guarantee
Fairness	Uniform clearing price within each batch; no ordering advantage
Determinism	All consensus-critical computations are pure functions of on-chain state
Composability	EVM precompiles enable atomic multi-market and multi-protocol operations
Transparency	Proposer inclusion behavior is publicly auditable on-chain
Liveness	Oracle fallback ensures markets continue operating during price feed interruptions

---

12. Security Analysis

11.1 Attack Vectors and Defenses

Proposer order injection to manipulate clearing price

Under batch auctions, a proposer who injects a buy order raises the clearing price; injecting a sell order lowers it. However, since all participants — including the proposer — execute at the uniform clearing price, the manipulation profit is limited to marginal price shifts. Furthermore, all validators independently recompute clearing prices during proposal verification, so the proposer cannot fabricate results.

Selective order exclusion by proposer

Excluding a large buy order can depress the clearing price, allowing the proposer to buy cheaper. However, under batch auctions, the profit opportunity from marginal clearing price manipulation is limited. The monitoring system records exclusion patterns, and persistent anomalies are surfaced for governance action.

Oracle manipulation to trigger incorrect maker protection

If an attacker manipulates the oracle price, the protection band may incorrectly filter out legitimate resting orders, reducing liquidity. The oracle uses stake-weighted median aggregation, making manipulation as expensive as controlling more than one-third of staked value. Additionally, the minimum retained depth parameter ensures baseline liquidity is preserved even under oracle stress.

Protection band boundary arbitrage

Arbitrageurs can compute the exact boundary prices of the protection band and target resting orders just within range. This is an accepted design trade-off — activity within the protection band is considered legitimate price discovery. Governance can adjust band widths based on observed data.

EVM fairness bypass via smart contracts

All EVM orders must pass through the CLOB precompile, which enforces identical validation logic as native transactions. Smart contracts cannot directly modify CLOB module state or receive preferential treatment in the batch auction algorithm. The EVM layer does not constitute a bypass path for fairness mechanisms.

Batch auction manipulation via multi-order EVM transactions

A smart contract can submit multiple orders in a single transaction via the precompile. These orders participate in the batch auction alongside all other orders and execute at the uniform clearing price. Atomic multi-order submission provides execution atomicity (all-or-nothing batch entry) but confers no price advantage.

11.2 Determinism Summary

Mechanism	Input Source	Determinism
Batch auction clearing price	Pending queue + filtered resting orders	Deterministic: pure function; all validators produce identical results

Mechanism	Input Source	Determinism
Maker protection filtering	Oracle price + resting orders + on-chain parameters	Deterministic: all inputs are consensus state
Inclusion monitoring	Local observed-order sets	Non-consensus: does not affect block validation
EVM precompile execution	EVM transaction → precompile → CLOB module	Deterministic: EVM execution is deterministic; precompile results depend on module state

---

13. Token Economics

The SVP token is the native asset of SVP Chain, serving as the foundation for consensus security, governance, and protocol-wide economic alignment.

12.1 Token Utility

The SVP token fulfills three distinct functions within the protocol:

Function	Mechanism
Consensus security	Validators and delegators stake SVP to participate in CometBFT consensus. Staked SVP secures the network, powers the stake-weighted median oracle, and determines voting power for block production.
Governance	SVP holders vote on protocol parameters, market listings, fee tier adjustments, protection band configurations, and emergency proposals. Voting power is proportional to staked amount.
Fee payment	Gas fees for native and EVM transactions are denominated in SVP and distributed to validators. Trading fees (maker/taker) are collected in the settlement asset (e.g., USDC) but fee tier discounts are unlocked by SVP staking.

12.2 Supply and Distribution

Total supply: 1,000,000,000 SVP (one billion), fixed at genesis. No inflation, no burning — the supply is constant and fully predictable. SVP uses 18 decimal places (consistent with the EVM standard), where the smallest on-chain unit is 10^{-18} SVP.

Allocation	Share	Amount	Vesting
Ecosystem fund	30%	300,000,000	5% at TGE; remainder released linearly over 48 months
Core contributors & advisors	15%	150,000,000	12-month cliff, then linear over 24 months (fully vested at month 36)
Testnet & co-builder incentives	12%	120,000,000	15% at TGE; remainder released linearly over 12 months
Liquidity & market making	10%	100,000,000	Fully available at TGE (constrained by market-making agreements)
Community & airdrops	10%	100,000,000	10% at TGE; remainder released linearly over 24 months
Staking rewards	13%	130,000,000	Activated upon mainnet launch; released linearly over 48 months
Foundation reserve	10%	100,000,000	6-month cliff, then linear over 36 months (fully vested at month 42)

12.2.1 Phased Token Lifecycle

SVP follows a two-phase lifecycle that separates early ecosystem bootstrapping from mainnet operation:

Phase 1 — ERC-20 pre-mainnet distribution. SVP is initially issued as an ERC-20 token on Ethereum, enabling early distribution to ecosystem participants, co-builders, and market participants before the mainnet is live. This phase serves to bootstrap the community, reward testnet contributors, and establish trading liquidity — ensuring that when the mainnet launches, an active and aligned token holder base already exists.

Phase 2 — Mainnet migration. Upon mainnet launch, a 1:1 bridge enables holders to migrate ERC-20 SVP to native mainnet SVP. The migration proceeds in stages:

Stage	Timing	Description
Bridge deployment	Mainnet launch	Official bridge contract deployed
Open migration	Mainnet + 1-3 months	Users can swap ERC-20 SVP 1:1 for native mainnet SVP
CEX transition	Post-stabilization	Exchange partners switch to native SVP deposits and withdrawals
ERC-20 retirement	Long-term	ERC-20 version gradually retired; bridge channel preserved

The staking rewards pool (13%) activates upon mainnet launch, providing a direct incentive for migration — only native mainnet SVP can participate in staking and earn rewards.

12.2.2 Vesting and Circulating Supply

The vesting schedule is designed to limit initial circulating supply while ensuring sufficient liquidity for market formation.

TGE (Token Generation Event) circulating supply:

Allocation	TGE Release	Amount
Ecosystem fund	5%	15,000,000
Core contributors & advisors	0%	0
Testnet & co-builder incentives	15%	18,000,000
Liquidity & market making	100%	100,000,000
Community & airdrops	10%	10,000,000
Staking rewards	0%	0

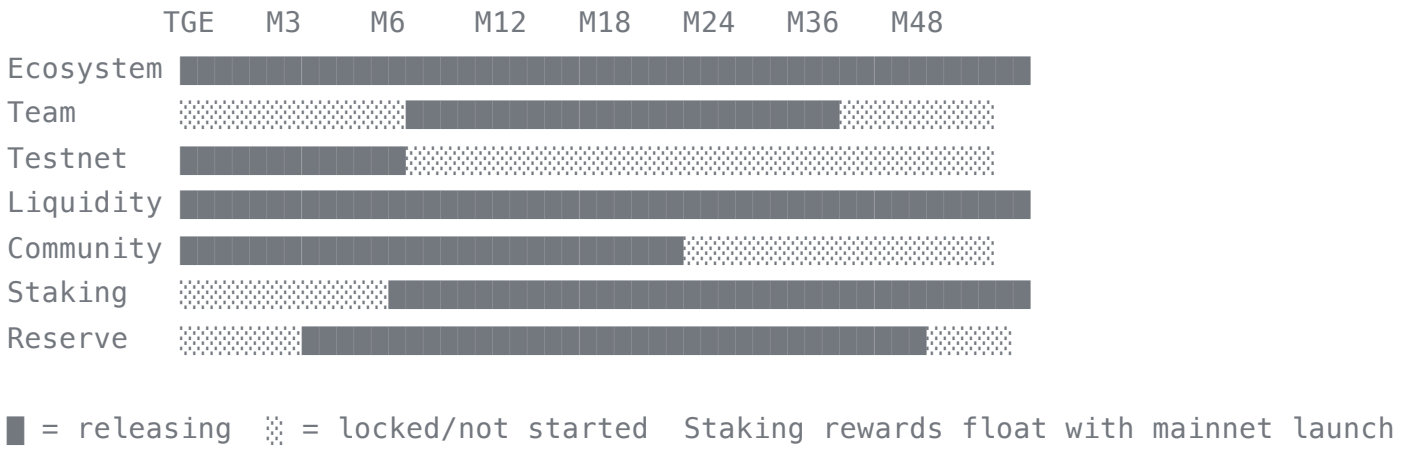
Allocation	TGE Release	Amount
Foundation reserve	0%	0
Total TGE circulation	**14.3%**	**143,000,000**

The liquidity & market making allocation, while fully unlocked, is operationally constrained by market-making agreements and is not freely circulating. Excluding this pool, free-float circulation at TGE is approximately 4.3% of total supply.

Monthly release schedule (first 13 months):

Period	Monthly New Release	Cumulative Circulation
TGE	143,000,000	14.3%
Month 1-6	~11,800,000/month	~21.4% at M6
Month 7-12	~12,100,000/month	~28.6% at M12
Month 13+	~18,700,000/month	Step-up from team unlock

The step-up at month 13 reflects the core contributor cliff ending — team tokens begin linear vesting, adding approximately 6,250,000 SVP per month to the release schedule.



12.2.3 Ecosystem Fund Deployment

The ecosystem fund (30%, 300M SVP) is the primary instrument for long-term ecosystem growth, deployed across the following categories:

Category	Allocation	Purpose
DEX liquidity mining	60M SVP (20%)	Incentivize liquidity providers across SVP trading pairs
Bridge incentives	30M SVP (10%)	Reward cross-chain bridge liquidity providers and early bridge users
Validator & node subsidies	45M SVP (15%)	Supplement staking rewards for early mainnet validators
Developer grants	45M SVP (15%)	Fund developers building applications on SVP Chain
Partnership integrations	30M SVP (10%)	Attract third-party projects to deploy on SVP Chain
Mainnet migration incentives	30M SVP (10%)	Bonus rewards for ERC-20 → mainnet SVP migration
Flexible reserve	60M SVP (20%)	Governance-directed deployment for future needs

12.2.4 Testnet Co-Builder Incentive Deployment

The testnet & co-builder allocation (12%, 120M SVP) rewards early contributors who help build and validate the network before mainnet:

Category	Allocation	Distribution Method
Testnet node operators	30M SVP (25%)	Based on uptime and stability
Testnet active users	30M SVP (25%)	Airdrop to active addresses (faucet, DEX, bridge usage)
Bug bounty	18M SVP (15%)	Rewards for valid bug submissions

Category	Allocation	Distribution Method
Developer contributions	18M SVP (15%)	Code contributions, documentation, tutorials
Content creators	12M SVP (10%)	Review articles, video tutorials
Flexible reserve	12M SVP (10%)	Future community needs

12.2.5 Liquidity Bootstrapping

The liquidity & market making allocation (10%, 100M SVP) is deployed at TGE to establish trading infrastructure:

- **CEX liquidity:** Initial token and settlement asset deposits required by exchange partners for order book market making
- **Market maker lending pool:** SVP made available to professional market makers under lending agreements to ensure continuous quote availability and tight spreads
- **DEX liquidity pools:** Future deployment into decentralized exchange pools to establish SVP/USDC on-chain trading liquidity, enabling participants to acquire SVP for gas fees and staking

Although fully unlocked at TGE, these tokens are operationally constrained by market-making agreements and are not freely tradeable.

12.3 Fee Structure and Revenue Distribution

SVP Chain generates revenue from two fee layers:

Trading fees are collected on every fill in the settlement asset:

Tier	30-Day Volume	Maker Fee	Taker Fee
Base	< $1M	0.02%	0.05%
Silver	$1M - $10M	0.015%	0.04%
Gold	$10M - $100M	0.01%	0.035%

Tier	30-Day Volume	Maker Fee	Taker Fee
Platinum	> $100M	0.005%	0.025%

Staking SVP unlocks accelerated tier progression — a trader staking above a threshold qualifies for one tier above their volume-based tier, incentivizing long-term token commitment.

Gas fees are collected in SVP on all transactions (native and EVM) and distributed to validators as block rewards. This provides validators with a direct SVP-denominated income stream in addition to their share of trading fee revenue.

Trading fee distribution:

```
Trading fees collected (USDC)
        │
        ├── 40%  → Staking rewards (distributed to validators and delegators)
        ├── 25%  → Insurance fund (allocated to per-market isolated funds)
        ├── 20%  → Protocol treasury (governance-controlled)
        ├── 10%  → Affiliate rewards (referral program)
        └──  5%  → Vault depositor incentives
```

This distribution ensures that the majority of protocol revenue flows to participants who contribute to network security (stakers) and system resilience (insurance), with governance retaining flexibility through treasury allocation.

12.4 Staking Economics

Validator requirements: Validators must stake a minimum SVP amount (governance-configurable) and operate both a full node and a Slinky oracle sidecar. Validators who fail to submit timely oracle price reports face reduced rewards, aligning the staking incentive with oracle reliability.

Delegator participation: SVP holders who do not operate validators can delegate their stake to active validators, earning a share of staking rewards proportional to their delegation minus the validator's commission rate.

Unbonding period: 21 days. This ensures that stakers cannot rapidly exit to avoid slashing or to front-run governance decisions. The unbonding period also supports the oracle security model — a validator cannot manipulate prices and immediately unstake to avoid consequences.

Slashing conditions:

- Double signing: 5% of staked SVP slashed

- Extended downtime (>24 hours): 0.1% slashed
- Oracle liveness failure (persistent non-reporting): reduced reward allocation (no principal slashing)

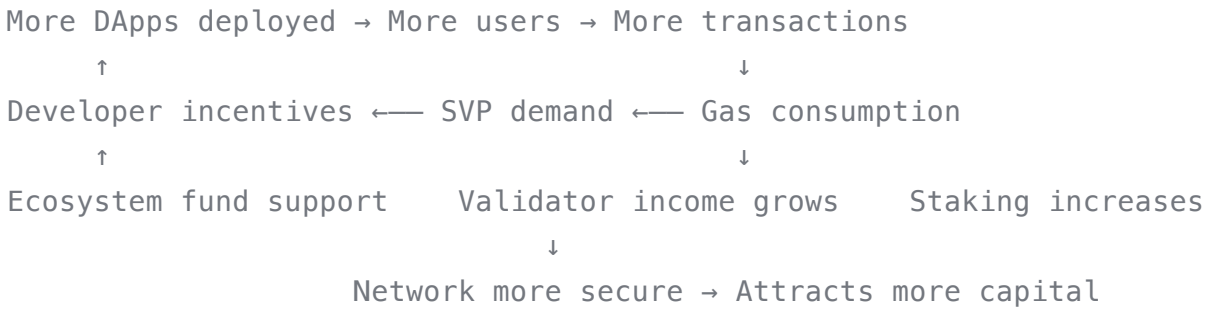
Staking reward sources: Stakers earn from two streams:

Source	Currency	Behavior
Staking rewards pool	SVP	Fixed pool: 130M SVP released linearly over 48 months post-mainnet (~2.7M/month)
Trading fee share	USDC	Variable: 40% of trading fees distributed proportionally to stakers
Gas fees	SVP	Variable: all gas fees distributed to validators

The staking rewards pool provides predictable SVP-denominated yield during the platform's growth phase. As the pool is gradually distributed, trading fee revenue (in USDC) is expected to grow as a share of total staking income, providing a stable-asset yield tied to platform adoption.

12.5 Value Accrual

The SVP token accrues value through a reinforcing flywheel:



Value Capture Mechanism	Description
Gas fee consumption	Every on-chain transaction consumes SVP as gas, creating continuous demand proportional to network activity
Staking lockup	Validators and delegators lock SVP through staking, reducing circulating supply; 21-day unbonding period ensures sustained supply

Value Capture Mechanism	Description
	reduction
LP lockup	Liquidity providers lock SVP in DEX pools and bridge contracts
Governance participation	Governance voting may require locking SVP
Bridge fees	Cross-chain operations consume SVP
Ecosystem project demand	Projects deployed on SVP Chain require SVP for gas
Fee tier utility	Staking SVP unlocks lower trading fees, creating organic demand from active traders

12.6 Insurance Fund Capitalization

Each perpetual market's insurance fund is seeded from two sources:

- **Ongoing revenue:** 25% of trading fees from each market flow into that market's isolated insurance fund
- **Liquidation surplus:** When liquidation proceeds exceed the deficit (the liquidated position's collateral covers more than the loss), the surplus accrues to the insurance fund

Insurance funds are denominated in the settlement asset (e.g., USDC), not in SVP, ensuring that fund value is not subject to token price volatility.

12.7 Economic Risks and Mitigations

Risk	Description	Mitigation
ERC-20 contract security	Smart contract vulnerabilities in the ERC-20 token	Third-party audit before launch; on-chain vesting contracts (e.g., Sablier) for programmatic enforcement of unlock schedules
ERC-20 migration risk	Incomplete migration leaves fragmented	Staking rewards activate only on mainnet, creating strong migration incentive; CEX partners

Risk	Description	Mitigation
	token supply	transition to native SVP deposits; bridge remains available for late migrators
Staking reward exhaustion	13% staking pool fully distributed after 48 months	Ecosystem fund validator subsidies (15% of 300M) provide supplementary rewards; trading fee revenue expected to grow as primary staking income by year 4; governance can redirect treasury funds if needed
Low-volume periods	Insufficient fee revenue reduces USDC staking yield	Staking rewards pool provides SVP-denominated baseline yield during growth phase; ecosystem fund subsidies available; foundation reserve deployable via governance
Insurance fund depletion	Extreme market event exhausts a market's insurance fund	Per-market isolation prevents cross-contamination; socialized loss mechanism (pro-rata haircuts to profitable positions) as last resort; governance can authorize treasury replenishment
Regulatory compliance	Token issuance subject to evolving regulatory requirements	Issuing entity structured in compliant jurisdiction; vesting enforced on-chain; transparent allocation and distribution

14. Governance Parameters

Parameter	Recommended Initial Value	Description
Batch auction window	1 block (~1-2 seconds)	Equal to block time; simplest implementation
`OracleProtectionBandPpm` (BTC)	3,000 (0.3%)	Lower volatility

Parameter	Recommended Initial Value	Description
`OracleProtectionBandPpm` (ETH)	5,000 (0.5%)	Moderate volatility
`OracleProtectionBandPpm` (small-cap)	10,000 (1.0%)	Higher volatility
Minimum retained depth	5 levels	Minimum price levels preserved regardless of protection band filtering
Monitoring report epoch	1,000 blocks	Approximately 20-30 minutes per reporting period
Marginal fill allocation	Pro-rata	New orders at the marginal price level allocated proportionally by quantity

---

15. Limitations and Future Work

14.1 Known Limitations

- **Cross-block MEV is not fully eliminated.** Batch auctions eliminate intra-block MEV, but information advantages across blocks (e.g., advance knowledge of upcoming oracle price updates) persist. Maker protection mitigates but does not fully resolve this.
- **Oracle dependency.** The maker protection mechanism's effectiveness depends on oracle accuracy and timeliness. Oracle failure or manipulation degrades protection quality.
- **Monitoring is non-enforcing.** Inclusion monitoring relies on community governance response and cannot automatically prevent malicious behavior in real time.
- **Batch auction latency.** Orders are no longer matched instantaneously; they must wait for the current block's batch auction to execute. This is an intentional trade-off of speed for fairness.

14.2 Future Directions

- **Commit-reveal scheme.** If monitoring data indicates that proposer manipulation remains significant, a lightweight commit-reveal mechanism can be introduced as a complementary

measure. A bond mechanism would be required to address the strategic non-reveal (free option) problem.

- **MEV redistribution.** If cross-block arbitrage impact on market makers is quantified as significant, mechanisms for capturing and redistributing extracted value to affected traders can be explored.
- **Cross-source oracle diversification.** Incorporate non-exchange price sources — such as OTC desk indicative quotes, options-implied spot prices, or decentralized oracle networks — as additional oracle inputs, reducing reliance on a small set of centralized exchanges and mitigating circular dependency risk as SVP Chain's own order flow grows.
- **Decentralized proposer selection.** Randomized or rotation-based proposer selection mechanisms to reduce any single proposer's sustained manipulation capability.

---

16. Conclusion

SVP Chain introduces a fundamentally fair trading architecture for on-chain perpetual futures through three complementary mechanisms:

1. **Batch auctions** eliminate order-sequence sensitivity, rendering front-running, sandwich attacks, and latency arbitrage economically unviable.
2. **Oracle-based maker protection** reduces adverse selection losses for liquidity providers, lowering the cost of market making and ultimately tightening spreads for all traders.
3. **Proposer inclusion monitoring** provides transparency and deterrence against residual manipulation vectors, while avoiding false punishment of honest validators in an asynchronous network.

The EVM compatibility layer, implemented through precompiled contracts, opens the fair trading system to the Ethereum ecosystem without introducing fairness gaps — enabling smart contract composability and toolchain compatibility while ensuring all transactions receive identical treatment.

The architecture maintains a deliberate balance between consensus determinism, computational feasibility, and governance configurability. Intra-block parallel execution, enabled by the natural independence of per-market batch auctions, provides a concrete scalability path as the platform grows.

SVP Chain demonstrates that fairness and performance are not opposing goals. By redesigning the matching engine around batch auctions rather than continuous execution, the economic incentives

for predatory trading behavior are removed at the protocol level — creating a trading environment where participants compete on the quality of their trading decisions, not the speed of their infrastructure.