Travix Protocol: Verifiable Order Matching, Probabilistic Event Trading, and On-Chain Proof of Reasoning

Travix Labs

June 2026

Abstract

Decentralized derivatives infrastructure faces four structural deficiencies: asset isolation between crypto and real-world markets, an asymmetric tool gap between institutional and retail participants, information latency preventing prediction market signals from being transacted at machine speed, and opacity of AI-driven trading decisions. This paper introduces the Travix Protocol, a sovereign application chain built on Cosmos SDK with a ZK-Validium verification layer. The architecture comprises four pillars. (1) *TravixCore*: a sovereign chain implementing dual-finality semantics—millisecond soft confirmations and minute-level hard finality via Groth16 proofs—with a novel margin-aware Order Book Tree (TravixOBT) achieving $\Theta(\log N)$ complexity for all operations, native privacy via a TEE+ZK hybrid Shielded Order Protocol, and a Verifiable AI Attestation Layer (Proof of Reasoning). (2) *Travix AI Agent Infrastructure*: a cognitive-execution decoupled architecture with a Regime-Adaptive Strategy Manifold that formally separates static strategy logic from dynamic market-regime-conditioned parameters, enabling AI-generated strategies whose risk controls adapt in real time to implied volatility, funding rates, and cross-asset correlations. (3) *Oracle Terminal*: an information-theoretic event trading engine that models prediction market probability changes as Kullback–Leibler surprises propagated through a calibrated transmission network with exponential temporal decay and Bayesian cross-event dependencies, producing asset-level force field vectors for automated position management. We present formal definitions, complexity analyses, and mathematical frameworks establishing these as original contributions to the intersection of verifiable computation, AI transparency, and event-driven finance.

1. Introduction

The global derivatives market exceeds $1.5 quadrillion in notional value, yet decentralized protocols capture less than 3% of crypto derivatives volume alone. Recent advances in on-chain order book exchanges—Hyperliquid [1], dYdX [2], Lighter [3]—have demonstrated that decentralized infrastructure can approach centralized-exchange performance in throughput and latency. However, these protocols solve a narrow problem: matching buy and sell orders for a limited set of crypto-native perpetual contracts. Four structural deficiencies prevent decentralized derivatives from becoming a comprehensive financial infrastructure layer.

Asset Isolation. When a Federal Reserve rate decision simultaneously moves U.S. Treasury yields, the dollar index (DXY), gold, crude oil, and Bitcoin, a trader must coordinate positions across three to five fragmented venues: a traditional brokerage for commodities, a centralized crypto exchange for BTC perpetuals, and one or more on-chain DEXs for DeFi-native assets. No existing decentralized exchange offers a unified margin account spanning crypto-native and real-world asset (RWA) perpetu-

als. The capital inefficiency is severe: a hedged portfolio requiring $50,000 in margin under a cross-asset model may demand $200,000 across isolated accounts, a 4× capital penalty. This fragmentation is not merely inconvenient—it structurally prevents the construction of macro-driven, multi-asset strategies that institutional participants consider table stakes.

Tool Asymmetry. Institutional market makers deploy sophisticated quantitative frameworks—CTA momentum models, dynamic grid engines, cross-venue funding rate arbitrage—with parameters that adapt continuously to market regimes: implied volatility shifts, correlation breakdowns, and liquidity regime changes. When the VIX spikes from 15 to 35, these systems automatically widen stop-losses, reduce position sizes, and shift from mean-reversion to momentum strategies. Retail participants and AI agents alike are confined to static parameter configurations—fixed 2% stop-losses, fixed grid spacings, fixed leverage—that fail catastrophically during regime transitions. The result is a structural transfer of wealth from static participants to adaptive ones. This asymmetry is not a matter of information access but of *tool infrastructure*: the mathematical frameworks for regime-adaptive parameter conditioning exist in academic literature [10] but have never been made available as composable, deployable primitives on a decentralized exchange.

Information Latency. Prediction markets such as Polymarket [4] and Kalshi have emerged as remarkably accurate real-time forecasting instruments, aggregating millions of dollars in risk-bearing capital to produce probability estimates that frequently lead traditional asset markets by minutes to hours. During the 2024 U.S. presidential election, prediction market prices began reflecting outcome probabilities hours before major news networks called results, and correlated assets (DXY, BTC, equities) moved in the direction implied by prediction market signals with measurable delay. Yet no existing exchange infrastructure systematically translates the *information content* of these probability changes —measured not as raw percentage shifts but as information-theoretic surprises—into executable trading directives at machine speed. The information is publicly available; the infrastructure to act on it algorithmically does not exist.

AI Opacity. Autonomous AI agents are increasingly driving trading decisions, from simple grid bots to LLM-powered macro analysis systems. Yet users who delegate capital to these agents lack any mechanism to verify that the AI performed the analysis it claims. Did the agent actually analyze the Fed minutes before opening a short position, or did it hallucinate a rationale post-hoc? The reasoning process remains an opaque black box, fundamentally contradicting the "Don't Trust, Verify" principle upon which decentralized finance is built. As AI-driven trading volume grows—estimates suggest autonomous agents will generate over 50% of on-chain trading volume by 2027—this opacity becomes a systemic risk rather than a mere inconvenience.

1.1. Why Now: Convergence of Enabling Technologies

Three technological developments have matured simultaneously, making a comprehensive solution feasible for the first time. First, *zero-knowledge proof systems* have achieved practical performance: Groth16 provers can generate proofs over circuits with billions of constraints in minutes rather than hours, and on-chain verification costs have dropped to acceptable levels on modern L1s [15]. Second, *large language models* have reached sufficient capability to generate, evaluate, and iterate on quanti-

tative trading strategies from natural language specifications, enabling a new paradigm where strategy creation is accessible to non-programmers while maintaining mathematical rigor through compiled execution [9]. Third, *prediction markets* have crossed the liquidity threshold necessary for their probability signals to be treated as reliable information sources, with Polymarket alone processing billions in cumulative volume across hundreds of active contracts [4]. The convergence of verifiable computation, AI cognition, and real-money probability signals creates an opportunity to build financial infrastructure that is simultaneously more powerful, more transparent, and more accessible than any existing alternative.

1.2. Contributions

This paper introduces the Travix Protocol, a sovereign application chain addressing all four structural deficiencies through an integrated architecture comprising four pillars. The principal contributions are:

1. **TravixOBT (Margin-Aware Order Book Tree):** A novel data structure extending prefix-tree indexing with six-value internal node aggregation—including, uniquely, margin requirement sums—achieving $\Theta(\log N)$ complexity for all operations: matching, insertion, quote retrieval, and margin verification. The margin verification improvement from $O(N)$ to $\Theta(\log N)$ is, to our knowledge, the first such result in the verifiable order book literature.
2. **Dual-Finality ZK-Validium:** A Cosmos SDK sovereign chain implementing millisecond soft confirmations and minute-level hard finality via Groth16 proofs, with multi-layered circuit aggregation (block → segment → batch) and a governance-managed circuit lifecycle enabling zero-downtime upgrades.
3. **Shielded Order Protocol:** A TEE+ZK hybrid privacy scheme combining Pedersen commitments for order concealment with Groth16 commitment-linkage proofs for post-match verification, achieving both execution-speed privacy and cryptographic correctness guarantees.
4. **Regime-Adaptive Strategy Manifold:** A formal decomposition $S = L(\theta(\Phi_t))$ that mathematically separates static strategy logic from dynamic market-regime-conditioned parameters, with Hidden Markov Model regime detection driving real-time parameter adaptation. This framework transforms AI-generated strategies from fragile static configurations into robust, self-adapting systems.
5. **Information-Theoretic Event Field:** An original framework modeling prediction market probability changes as Kullback–Leibler divergence surprises propagated through a calibrated transmission network with exponential temporal decay kernels and Bayesian cross-event conditioning, producing asset-level force field vectors for automated position management.
6. **Proof of Reasoning (PoR):** A Verifiable AI Attestation protocol committing cryptographic digests of AI decision artifacts—inputs, outputs, model identifiers, reasoning traces—to the chain via Merkle trees co-verified within the Groth16 batch proof, providing, to our knowledge, the first systematic approach to on-chain AI decision auditability in a production trading system.

1.3. Paper Organization

Section 2 presents the architecture overview and introduces the AI–Verifiable–Private trilemma. Section 3 describes TravixCore—the sovereign verifiable chain—in depth, covering the state tree, TravixOBT, the deterministic parallel transaction engine, ZK-Validium verification, matching engine, liquidation system, privacy infrastructure, Proof of Reasoning, and escape hatch mechanisms. Section 4 details the AI Agent infrastructure, including the cognitive-execution decoupled architecture, regime-adaptive strategy manifold, and clone engine. Section 5 presents the Oracle Terminal and its information-theoretic event trading framework. Section 6 concludes with a discussion of limitations and future work.

2. Architecture Overview

2.1. The Travix Stack

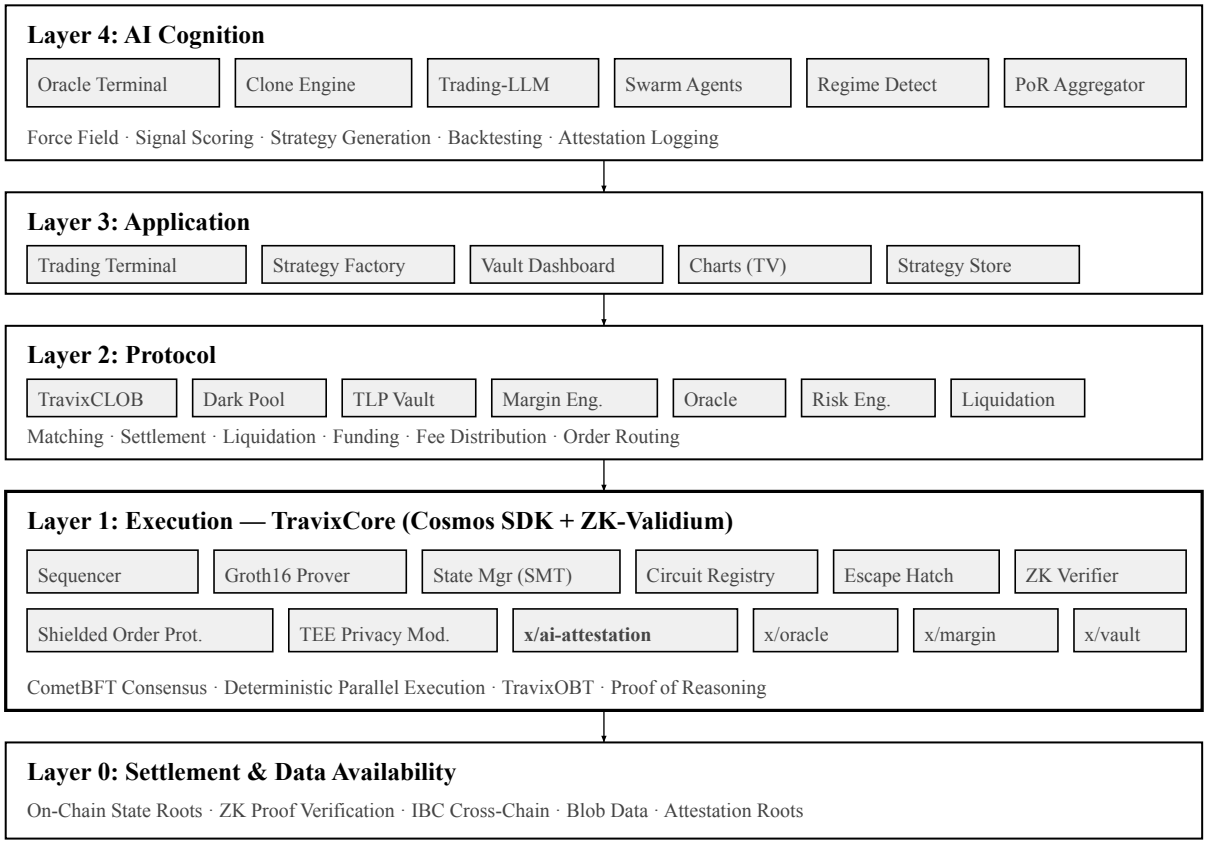

Figure 1: The Travix Stack. Bold border denotes the trust boundary (Layer 1). The `x/ai-attestation` module anchors AI decision transparency to the chain's cryptographic state.

2.2. Transaction Lifecycle

A transaction traverses seven deterministic stages: (1) Signal generation (AI or human); (2) Signing via Session Key or private key; (3) Sequencing into Fast Lane or Slow Lane; (4) Matching in the parallel execution engine; (5) Soft finality via `MsgSubmitBatch` (state: `SUBMITTED`); (6) Groth16 proof generation (asynchronous, includes PoR attestation root); (7) Hard finality via `MsgSubmitProof` (state: `CONFIRMED`, cryptographically irreversible). AI-originated trades additionally generate attestation artifacts committed alongside trade data in stage 6.

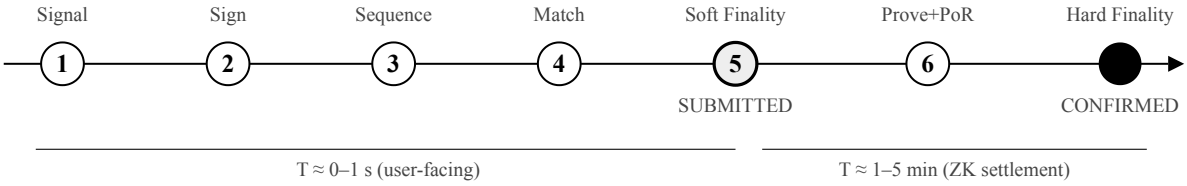

Figure 2: Transaction lifecycle with dual-finality semantics.

2.3. The AI–Verifiable–Private Trilemma

> ***Definition 1 (AI–Verifiable–Private Trilemma).*** A derivatives protocol seeks three properties: (*A*) AI-Native—the protocol integrates ML inference into execution; (*V*) Verifiable—all state transitions including AI decisions are provably correct or auditable; (*P*) Private—order parameters are concealed. Existing protocols achieve at most two. Travix resolves the trilemma through layered separation: AI at Layer 4 with on-chain attestation (PoR), verifiability at Layer 1 via Groth16, privacy at Layer 1 via TEE+ZK shielded orders.

3. TravixCore: Sovereign Verifiable Chain

3.1. Cosmos SDK Sovereign Application Chain

TravixCore is implemented on Cosmos SDK with CometBFT consensus [5], providing sovereign control over block time, fee model, validator set, and upgrade schedule with IBC cross-chain interoperability. Trading logic is implemented as protocol-level modules with direct state machine access, avoiding EVM overhead.

Table 1: TravixCore module architecture.

Module	Function	Layer
`x/matching`	CLOB engine with price-time priority, TravixOBT	Core
`x/settlement`	Position settlement, PnL, fee distribution	Core
`x/margin`	Isolated/cross/portfolio margin, liquidation triggers	Core
`x/zk-settlement`	Optimistic batch submission, Groth16 proof verification	ZK
`x/exit-hatch`	Forced withdrawal via Merkle proof	ZK
`x/circuit-registry`	Circuit version management, VK storage	ZK
`x/privacy`	Shielded order commitments, TEE attestation	Privacy
`x/ai-attestation`	PoR: AI artifact Merkle roots, verification queries	AI
`x/oracle`	Multi-source price aggregation, RWA pricing	Infra
`x/vault`	TLP vault, LP share accounting	DeFi

3.2. State Tree and TravixOBT

TravixCore maintains state within a Sparse Merkle Tree (SMT) using the Poseidon2 hash function [6], structured around domain-specific sub-trees.

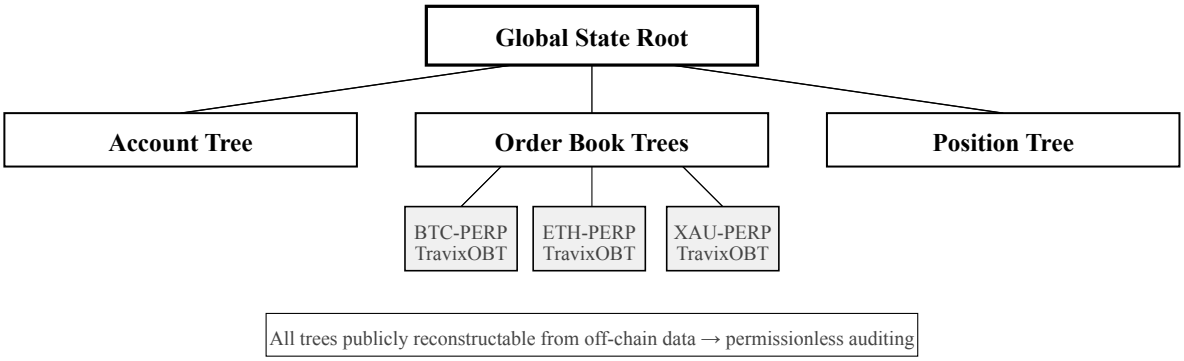

Figure 3: State Tree structure. Each market uses a TravixOBT instance (§3.2.2).

3.2.1. Account Tree

The Account Tree stores per-user data: asset balances, position metadata, sub-account mappings, API key nonces with Dead Man's Switch timestamps, and public pool share information. Each leaf contains sufficient data for Escape Hatch exit proofs.

3.2.2. TravixOBT: Margin-Aware Order Book Tree

We introduce the *TravixOBT*, a specialized data structure extending prefix-tree indexing with margin-aware aggregate internal nodes. For a price field of P bits and nonce field of O bits, tree height $H = P$

$+ O$.

Definition 2 (TravixOBT Leaf Index). For an ask order with price p and nonce o:

$$I_{ask} = p \times 2^O + o \tag{1}$$

For a bid order with price p and nonce o:

$$I_{bid} = p \times 2^O + (2^O - 1 - o) \tag{2}$$

Lemma 1 (Index-Priority Equivalence). For two ask orders, $I_{\text{order}_\text{i}} < I_{\text{order}_\text{j}}$ iff $\text{price}_\text{i} < \text{price}_\text{j}$, or ($\text{price}_\text{i} = \text{price}_\text{j}$ and $\text{nonce}_\text{i} < \text{nonce}_\text{j}$). A smaller index indicates higher priority for asks. Symmetrically, for bids, a larger index indicates higher priority. The tree's leaf ordering exactly encodes price-time priority.

Proof sketch. The upper P bits of I encode price; lower O bits encode nonce (or complement for bids). Integer comparison on I lexicographically compares (price, nonce). □

Each internal node i stores six aggregated values from its sub-tree. The first four follow standard practice [3]:

$$\text{AskSizeSum}_i = \sum_{j \in SubTree_i} size_j \times isAsk_j \tag{3}$$

$$\text{BidSizeSum}_i = \sum_{j \in SubTree_i} size_j \times (1 - isAsk_j) \tag{4}$$

$$\text{AskQuoteSum}_i = \sum_j size_j \times price_j \times isAsk_j \tag{5}$$

$$\text{BidQuoteSum}_i = \sum_j size_j \times price_j \times (1 - isAsk_j) \tag{6}$$

TravixOBT extends this with two margin-aware aggregates. These encode *initial margin requirements under Isolated Margin mode*, where each order's margin is a static function of its entry price and leverage—independent of mark price fluctuations and cross-asset correlations:

$$\text{AskMarginReq}_i = \sum_j (size_j \times price_j / leverage_j) \times isAsk_j \tag{7}$$

$$\text{BidMarginReq}_i = \sum_j (size_j \times price_j / leverage_j) \times (1 - isAsk_j) \tag{8}$$

All six values satisfy recursive child-to-parent relations and propagate along a single root-to-leaf path upon modification: $\Theta(\log N)$ updates. The hash of each node incorporates both commitment and aggregates:

$$\text{Hash}(i) = \text{Poseidon2}(\text{Hash}(left_i), \text{Hash}(right_i), \text{InternalData}_i) \tag{9}$$

Table 2: Complexity comparison.

Operation	TravixOBT	Embedded LL	Naive LL
Existence proof	$\Theta(\log N)$	$\Theta(\log N)$	$\Theta(N)$
Best ask/bid	$\Theta(\log N)$	$\Theta(\log N)$	$\Theta(1)$
Quote retrieval	$\Theta(\log N)$	$O(N \log N)$	$\Theta(N)$
Margin verification	$\mathbf{\Theta(\log N)}$	$O(N)$	$O(N)$
Total hashes per cycle	$\mathbf{\Theta(\log N)}$	$O(N \log N)$	$\Theta(N)$

The margin verification row is the key differentiator: by storing MarginReq aggregates, the circuit verifies total margin commitment in a single path traversal rather than iterating over all active orders.

3.3. ZK-Validium Verification Layer

3.3.1. Dual-Finality Model

Definition 3 (Dual Finality). *Soft Finality* (T≈0): Sequencer commits batch via `MsgSubmitBatch` → `SUBMITTED`. Trust: honest Sequencer. *Hard Finality* (T≈2–10 min): STARK-then-Groth16 hybrid proof π verified on-chain → `CONFIRMED`. Trust: Groth16 soundness over BN254. If proof not submitted within `ProofTimeout` (24h), Escape Hatch activates (§3.9).

3.3.2. Multi-Layered Circuit Architecture

Raw Groth16 proving over billions of constraints is computationally infeasible for production latency targets. The Travix Prover therefore implements a *STARK-then-Groth16 hybrid pipeline*: inner proofs at the block and segment level use FRI-based STARKs (prover-friendly, parallelizable, no trusted setup), while a final Groth16/PLONK wrapper compresses the aggregated STARK proof into a constant-size proof (~200 bytes) for on-chain verification. This architecture mirrors production-deployed systems such as Polygon zkEVM and zkSync Era [15].

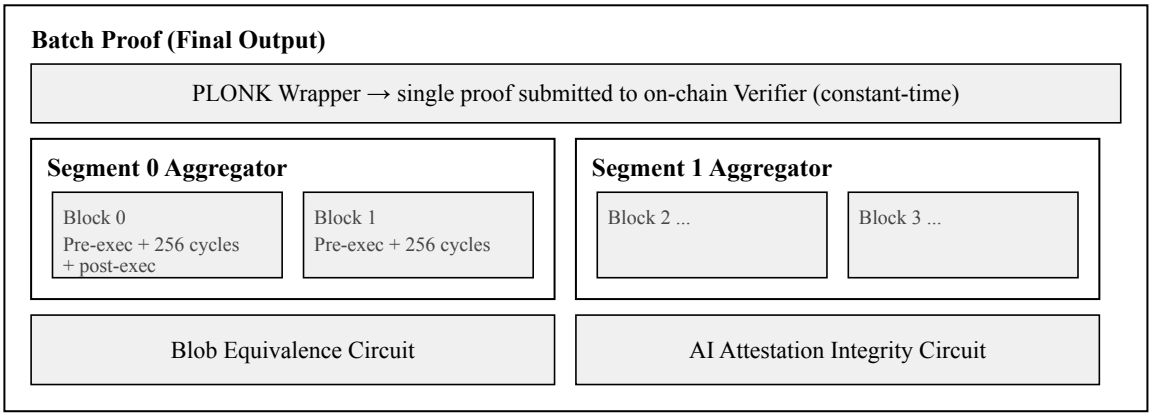

Figure 4: Multi-layered proof aggregation. Block proofs aggregate into segments, segments into a single batch proof wrapped in PLONK. The Attestation Integrity Circuit verifies the PoR Merkle root.

Pre-Execution updates global market state: applying funding payments when the block timestamp exceeds the funding interval, updating index and mark prices from oracle feeds, computing market premiums, and recalculating mark prices with EMA-capped deviation bounds.

Execution Cycle processes one instruction per cycle. For a new transaction, the circuit verifies the signature against the API key stored in the Account Tree, checks the nonce for replay protection, and validates witness data by verifying Merkle paths against the State Root. Risk checks confirm margin sufficiency using the MarginReq aggregates in TravixOBT. The circuit then executes based on instruction type—order insertion, pending trade execution, order cancellation, or liquidation—updates affected leaf data, recomputes internal node aggregates along the modified path, and derives the new State Root.

Crossing Detection. When inserting a new ask order at index I with bit decomposition $I = I_0 ... I_{H-1}$, the total crossing bid size is computed by traversing the root-to-leaf path $L = \{L_i\}$:

$$crossingSize = \sum_{L_i \in L,\, 0<i\leq H} (\text{BidSizeSum}_{L_i} - \text{BidSizeSum}_{L_{i-1}}) \times (1 - I_{i-1}) \tag{10}$$

If crossingSize $= 0$, no matching bids exist and the order is inserted. This requires only $\Theta(\log N)$ operations along a single path.

Table 3: Constraint complexity per execution cycle (calibrated).

Component	**Constraints**
Signature verification (Ed25519)	~30,000
Merkle path verification (Poseidon2, depth 32)	~25,000
Order insertion + crossing check	~80,000
Trade execution + balance update	~60,000
Margin sufficiency (MarginReq aggregates)	~15,000
State root recomputation	~50,000
Total per cycle	**~260,000**

With 256 cycles/block, 4 blocks/segment, 8 segments/batch: ~8,192 cycles totaling ~2.1×10^9 constraints per batch. Direct Groth16 proving at this scale would require hours on commodity hardware. The hybrid pipeline resolves this: each block is proven independently as a STARK proof (~10s on GPU-accelerated provers, 32 blocks provable in parallel), segment aggregators recursively compose block STARKs (~30s per segment), and the final batch wrapper converts the aggregated STARK into a Groth16 proof over a verification circuit of ~2M constraints (~60s). Total proving latency: 2–10 minutes depending on batch density, with constant on-chain verification cost.

3.3.3. Groth16 Pipeline

Proposition 1 (Verifiable Matching Correctness). The Groth16 circuit proves: (i) Price-time priority for all matches; (ii) Volume constraints $q \leq \min(\text{buy.qty}, \text{sell.qty})$; (iii) Margin sufficiency via MarginReq aggregates; (iv) State transition integrity $R_{new} = F(R_{old}, B)$; (v) Attestation integrity: the PoR root A_{root} is included in public inputs.

On-chain verification checks the pairing equation:

$$e(A, B) = e(\alpha, \beta) \cdot e(\sum \gamma_i \cdot x_i, \gamma) \cdot e(C, \delta) \tag{11}$$

where x_i includes $\{R_{old}, R_{new}$, batch_hash, circuit_version, $A_{root}\}$.

3.3.4. Circuit Lifecycle Management

The `x/circuit-registry` module enables smooth upgrades via four phases: Registration (governance, ~7d) → Parallel Operation (~14d) → Deprecation (~7d) → Retirement.

3.4. Matching Engine and Order Semantics

3.4.1. Deterministic Parallel Transaction Engine (PTE)

Sequential transaction execution is the dominant bottleneck in on-chain trading systems: a batch of 10,000 transactions targeting 50 independent markets is processed serially even though 98% of transactions have disjoint state footprints. TravixCore introduces a Deterministic Parallel Transaction Engine (PTE) that extracts maximum parallelism from ordered transaction batches while preserving bit-exact determinism—a property essential for ZK verification.

3.4.2. Extended Access Lists and Dependency Analysis

Definition 4 (Extended Access List). Each transaction T_i carries an Extended Access List $\text{EAL}(T_i) = (R_i, W_i)$ declaring its read set R_i and write set W_i at the granularity of market-level state partitions. The EAL extends EIP-2930 access lists [18] with market-specific semantics: a BTC-PERP limit order declares $R = \{\text{AcctA, BTC-OB}\}$ and $W = \{\text{BTC-OB}\}$, where AcctA is the submitting account's state leaf and BTC-OB is the BTC-PERP order book sub-tree.

Definition 5 (Parallel Compatibility). Two transactions T_i, T_j (i<j) are parallel compatible iff their state footprints are disjoint:

$$(W_i \cap R_j = \varnothing) \wedge (R_i \cap W_j = \varnothing) \wedge (W_i \cap W_j = \varnothing) \tag{12}$$

The PTE Scheduler constructs a conflict graph $G = (V, E)$ from the ordered batch $B = (T_1, ..., T_n)$ where vertices are transactions and edges connect conflicting pairs. From G, the Scheduler derives a

Directed Acyclic Graph (DAG) encoding execution dependencies: $T_i \rightarrow T_j$ iff $i < j$ and $(T_i, T_j) \in E$. The DAG is partitioned into Execution Waves—maximal independent sets computed via topological layering of the conflict DAG.

3.4.3. Wave-Based Parallel Dispatch

Execution proceeds in three wave phases, each with distinct parallelism characteristics:

Wave 1: Non-Conflicting State Transitions (Parallel Phase). Transactions with disjoint state footprints execute simultaneously across isolated Market Actors. Each Market Actor is an independent execution context owning a single market's TravixOBT instance and associated state sub-tree. For a batch containing M active markets, up to M Market Actors execute concurrently. Within each Actor, transactions targeting the same market execute in sequence, preserving the Sequencer's ordering invariant.

Wave 2: Intra-Market Dependent Transitions (Serialized Phase). State-conflicting transactions within the same market—e.g., a BTC buy followed by a BTC cancel from the same account—execute sequentially within their respective Market Actor. The Scheduler ensures these are dispatched only after Wave 1 predecessors have committed their state deltas.

Wave 3: Global State Dependencies (Finalization Phase). Cross-market operations—global risk checks, funding rate settlements, portfolio margin recalculations, liquidation cascade evaluations—execute after all market-local waves complete. These operations read the aggregated state from multiple Market Actors and may trigger secondary transactions (liquidations, ADL) that are themselves parallelized via the same wave decomposition.

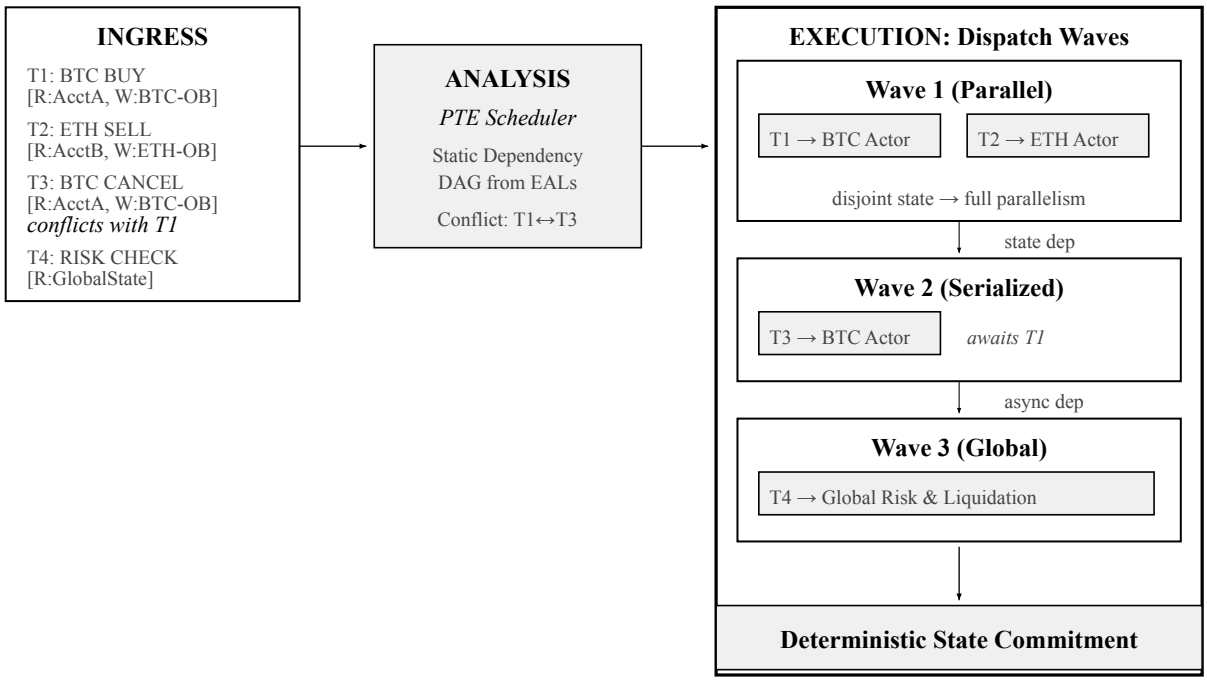

Figure 5: Deterministic PTE process flow. EAL-annotated transactions are analyzed for state conflicts, partitioned into dependency-respecting waves, and dispatched to isolated Market Actors. The conflict between T1 and T3 (shared BTC-OB write set) forces serialization in Wave 2, while T1 and T2 (disjoint markets) execute fully in parallel.

Table 4: PTE performance characteristics.

Metric	Serial Execution	PTE (50 markets)	Improvement
Throughput (tx/block)	256	~12,800	~50×
Wave 1 parallelism factor	1.0	$0.92 \times M$	Linear in markets
Intra-market conflict rate	N/A	~8%	Serialized in Wave 2
Global dependency overhead	N/A	<2%	Wave 3 finalization
State hash computation	$\Theta(N \log N)$	$\Theta((N/M) \log(N/M))$	$M\times$ parallel

Proposition 2 (Linear Scalability). For M markets with disjoint state, throughput $\mathrm{T}(M) = M \times \mathrm{T}(1)$. The serial fraction s in Amdahl's Law is bounded by the global dependency ratio: $s \leq |\text{Wave 3}| \,/\, |B|$, which empirically converges to <0.02 for production workloads. □

3.4.4. Parallel Merklization and State Convergence

After wave execution completes, disjoint state deltas from M Market Actors must be committed to the global Sparse Merkle Tree. Sequential Merklization would negate the parallelism gains of PTE. TravixCore implements three mechanisms for parallel state commitment:

(i) Independent Market Sub-Tries. Each market's state (TravixOBT, position data, market parameters) resides in a dedicated sub-trie of the global SMT. Since Market Actors produce disjoint state deltas by construction (enforced by EAL conflict analysis), sub-trie root hashes can be computed independently and concurrently across M cores.

(ii) Concurrent Sub-Tree Hashing. Within each sub-trie, the Poseidon2 hash computations along modified paths are parallelized via multi-core I/O. For a sub-trie with k modified leaves, the re-hashing cost is $\Theta(k \log(N/M))$, and independent paths hash concurrently.

(iii) Deterministic State Aggregation. The M updated sub-trie roots are assembled into the global state root in a fixed, canonical order (sorted by market ID). This aggregation is *idempotent*: the same ordered batch produces the same global root regardless of internal parallelism scheduling, enabling independent verification by Watcher nodes via full re-execution.

Proposition 3 (Parallel Merklization Complexity). For a batch modifying N total leaves across M disjoint sub-tries, the parallel hash computation cost is $\Theta((N/M) \cdot \log(N/M))$ with M-way parallelism, compared to $\Theta(N \cdot \log N)$ for serial computation. The global root aggregation adds $O(M)$ overhead. □

3.4.5. Host Function Interface and Inter-VM Communication

TravixCore implements a modular multi-VM execution environment. The primary execution runtime (TravixVM) handles high-performance trading operations—order matching, risk management, liquidation logic—as optimized Wasm modules within isolated Market Actors. A secondary EVM-compatible runtime (TravixEVM) provides standard smart contract functionality for DeFi composability: token standards, liquidity protocols, external integrations, and bridging.

The two runtimes communicate through a *Host Function Interface* (HFI): a deterministic inter-VM framework providing synchronous, atomic state access with zero-copy data passing [16]. When a TravixEVM contract needs to query order book state (e.g., for a DeFi protocol that references mark prices), it invokes an HFI host function that reads directly from the TravixVM's state sub-trie without serialization overhead. Conversely, TravixVM can invoke EVM precompiled contracts for vectorized risk computation and cryptographic primitives (pairing operations, hash functions) that benefit from hardware-optimized EVM implementations.

State confinement between VMs is enforced via EAL-style access lists (analogous to EIP-2930 [18]): each inter-VM call declares its cross-boundary state footprint, enabling the PTE Scheduler to treat inter-VM calls as standard transactions for conflict analysis. This ensures that multi-VM execution does not compromise the determinism guarantees required for ZK verification.

3.4.6. FlashCommit Protocol

> ***Definition 6 (FlashCommit Receipt).*** A signed receipt containing batch_id, tx_index, Hash(δ), sequencer_signature, and timestamp. Because PTE execution is deterministic given the ordered batch, the state_delta is a mathematically guaranteed prediction of final state, providing "economic finality" for strategy chaining. Any party with access to the batch data can independently verify the receipt by re-executing the PTE pipeline.

The Fast Lane handles orders, cancellations, and modifications with sub-millisecond time-triggered micro-batching. The Slow Lane handles deposits, withdrawals, and governance with physical CPU core isolation. Both lanes produce EAL-annotated transactions that enter the unified PTE pipeline.

3.5. Liquidation System

3.5.1. Margin Model

For each position, the initial margin rate IMR = 1/max_leverage and maintenance margin rate MMR = IMR × maintenance_factor (typically 0.5). Account health is defined as:

$$\text{Health} = (\text{Equity} - \textstyle\sum_i \text{MMR}_i \times |\text{Position}_i| \times \text{MarkPrice}_i) \,/\, \text{Equity} \tag{13}$$

Liquidation triggers when Health < 0. The Health computation is an Account Tree operation: Equity and unrealized PnL update with mark price changes, but the MarginReq aggregates cached in TravixOBT (Eqs. 7–8) remain stable because they encode initial margin at entry price—not mark-to-

market margin. This separation is critical: mark price updates affect Health via the Equity numerator (an O(1) account-level update), not via O(N) re-traversal of the order book tree.

Portfolio Margin. Under Portfolio Margin mode, correlated positions receive margin credit computed at the *Account Tree level*, not the Order Book Tree level. For an account holding P positions across distinct markets, the portfolio margin adjustment $\delta_{\text{portfolio}}$ is derived from the cross-asset correlation matrix Σ:

$$\delta_{portfolio} = \sum_{A \neq B} |\rho_{AB}| \cdot min(\text{MarginReq}_A, \text{MarginReq}_B) \cdot \text{sign\_offset}(A, B) \tag{13'}$$

where sign_offset is positive for offsetting (negatively correlated) positions. This O(P^2) computation (typically $P < 20$) produces an account-level margin credit that reduces the effective maintenance requirement in Eq. 13. The TravixOBT $\Theta(\log N)$ property is preserved: order book operations use the static Isolated Margin aggregates, while portfolio adjustments are computed as a separate account-level pass.

3.5.2. Progressive Liquidation

The liquidation engine operates as an asynchronous Market Actor executing a five-level cascade: (1) *Warning*: Health $<$ threshold$_1$, user notified; (2) *Cancel orders*: all open orders cancelled to free margin; (3) *Partial liquidation*: largest positions reduced with quantity liq_qty = min(|position|, margin_deficit / mark_price $\times$ safety_multiplier); (4) *TLP Backstop*: vault absorbs residual positions at a discount; (5) *Insurance Fund*: covers shortfall $\Delta = \max(0, \text{bankruptcy\_price} - \text{fill\_price}) \times \text{size}$.

3.5.3. Auto-Deleveraging (ADL)

When order book depth is insufficient for market-price liquidation, the system identifies counterparties via ADL priority ranking = profit_ratio $\times$ effective_leverage, sorted descending. ADL executes at the bankruptcy price rather than the mark price, ensuring no socialized losses. The liquidated position is matched against the highest-priority counterparty.

3.5.4. Insurance Fund

The insurance fund is sourced from partial liquidation penalties and protocol revenue. It covers losses when liquidation execution prices are worse than bankruptcy prices. The fund balance is transparently queryable on-chain.

3.6. Privacy-Preserving Infrastructure

3.6.1. Shielded Order Protocol

The protocol operates in three phases. In *Phase 1 (Commit)*, the user computes a Pedersen commitment [7] on the order O:

$$C = o \cdot G + r \cdot H \tag{14}$$

where (G, H) are independent curve generators, $o = \text{encode}(O)$, and r is random. ZK proofs π_1 (margin sufficiency) and π_2 (price range) are attached. In *Phase 2 (Match)*, the TEE enclave decrypts and matches at full speed. In *Phase 3 (Verify)*, the Prover generates π_3 (state transition) and π_4 (commitment linkage: fills correspond to submitted commitments).

3.6.2. TEE+ZK Hybrid Trust Model

The security model degrades gracefully across three failure modes, with an explicit separation between *correctness guarantees* (which depend only on ZK) and *privacy guarantees* (which depend on TEE):

(i) TEE intact. Full privacy (order parameters concealed from all parties including the Sequencer operator) and full speed (matching occurs on plaintext within the enclave). ZK proofs verify matching correctness and commitment linkage. This is the design-target operating mode.

(ii) TEE compromised (information leakage). An attacker with a TEE exploit observes order plaintext within the current batch window. This constitutes a *privacy degradation* but *not a correctness violation*: ZK proofs (Phase 3) still verify that all fills respect price-time priority, volume constraints, and margin sufficiency. The attacker gains MEV-style information advantage—the ability to front-run or sandwich orders within the batch interval—but cannot manipulate execution prices or steal funds. The privacy window of exposure is bounded by the batch interval (typically 2–10 minutes): an attacker observing orders within this window gains at most the information-theoretic advantage equivalent to observing the order book with batch-granularity delay, comparable to the latency advantage of co-located market makers in traditional finance.

(iii) TEE + Prover compromised. If both the TEE and the proof generation infrastructure are compromised, the on-chain verifier will reject invalid proofs (Groth16 soundness), and the Escape Hatch activates when proof submission exceeds `ProofTimeout`. Users prove balances via Merkle inclusion against the last valid state root.

The protocol adopts a defense-in-depth posture: SGX Remote Attestation verifies enclave integrity at session establishment, enclave memory is encrypted in hardware, and the batch-granularity privacy window limits the blast radius of any single vulnerability. We emphasize that the TEE assumption is the weakest component of the privacy model—SGX has been subject to multiple side-channel attacks historically [20]—and the protocol is designed such that TEE failure degrades only privacy, never correctness.

3.6.3. Dark Pool and Smart Order Router

The SOR minimizes total execution cost:

$$\min_{\alpha} C(\alpha) = \alpha \cdot C_{dark}(q\alpha) + (1-\alpha) \cdot C_{lit}(q(1-\alpha)) \tag{15}$$

3.7. Proof of Reasoning — Verifiable AI Attestation

Definition 7 (Proof of Reasoning). For an AI decision function f with input x and output $y = f(x)$, a PoR consists of a commitment $C_{PoR} = \text{Hash}(x \parallel y \parallel \text{model\_id})$ posted on-chain, such that any party with access to the off-chain log can verify C_{PoR}. This does not prove "AI computed correctly" (that requires zkML), but proves "AI used the data it claimed and produced the output it claimed."

The `x/ai-attestation` module stores Merkle roots of five artifact categories: (A1) Force Field Snapshots, (A2) LLM Reasoning Traces, (A3) Signal Score Breakdowns, (A4) Strategy Wasm Hashes, (A5) Execution Decision Logs. The attestation root is included as a public input in the Groth16 proof (Proposition 1, item v), so the validity proof simultaneously attests to trading correctness and AI log integrity.

The verification protocol: a user queries the on-chain attestation root for a given batch, retrieves the full log and Merkle proof from the Verifier Service, then locally verifies that Hash(log.input ∥ log.output ∥ log.model_id) equals the leaf hash, and that the Merkle proof links to the on-chain root. Future phases progress from hash commitment (current) to deterministic replay (open-source logic) to full zkML [8].

3.8. Oracle System

$$\text{Mark} = \text{Median}\big(\text{Impact}_{mid},\ \text{Oracle}_{median},\ \text{Last}_{trade}\big) \tag{16}$$

$$\text{Premium} = \text{Clamp}\big(\text{EMA}\big(\text{Mark} - \text{Index},\ \tau\big),\ -0.5\%,\ +0.5\%\big) \tag{17}$$

Funding rates computed hourly, TWAP premium plus interest, clamped $[-0.5\%, +0.5\%]$. For RWA off-hours, funding weight amplified by κ.

3.9. Escape Hatch

Activates when: a batch in `SUBMITTED` state exceeds ProofTimeout (24h), or a priority transaction is censored beyond CensorshipTimeout, or governance triggers emergency mode. Users prove balance via Merkle inclusion against the frozen state root, positions are settled at the last known mark price, and assets are transferred to the user's L1 address.

4. Travix AI Agent Infrastructure

4.1. Agent-Native Design Philosophy

Travix is designed as a dual-track exchange: humans interact via GUI trading terminals; AI agents interact via MCP protocol [9], structured Skills (JSON Schema), and direct API. Both tracks share a single unified liquidity pool, preventing fragmentation. The agent track provides 100+ machine-native Skills for querying market state, submitting orders, executing algorithms (TWAP), and managing risk —all optimized for LLM Function Calling.

4.2. Cognitive-Execution Decoupled Architecture

LLM inference requires 5–15 seconds while crypto markets move within milliseconds. Naively coupling AI cognition to order execution would introduce unacceptable latency: by the time an LLM completes its analysis, the trading opportunity has evaporated. The CED architecture resolves this fundamental tension by separating slow cognition from fast execution into two distinct infrastructure layers with an asynchronous compilation bridge between them.

The *Cloud Layer* operates on a timescale of seconds to minutes. It hosts LLM inference engines, strategy generation pipelines, backtesting infrastructure, and Wasm compilation toolchains. When a user requests a new strategy—either via natural language specification or by cloning historical behavior (§4.4)—the Cloud Layer performs multi-step reasoning: analyzing market structure, generating candidate strategy logic, backtesting across historical regimes, iterating on parameter bounds, and finally compiling the validated strategy into a deterministic Wasm module. This process may take 30–120 seconds but occurs *once*, producing an artifact that executes for days or weeks.

The *Edge Layer* operates on a timescale of microseconds to milliseconds. Compiled Wasm modules are deployed to Edge Nodes co-located with the Sequencer infrastructure, achieving sub-5ms tick-to-trade latency. Edge Nodes subscribe to real-time market data feeds (order book updates, trade prints, funding rate changes, Oracle Terminal force field vectors) and inject these as input parameters into the Wasm runtime at each tick. The Wasm module executes deterministically: given the same inputs, it produces the same outputs on any machine, a property essential for PoR verification.

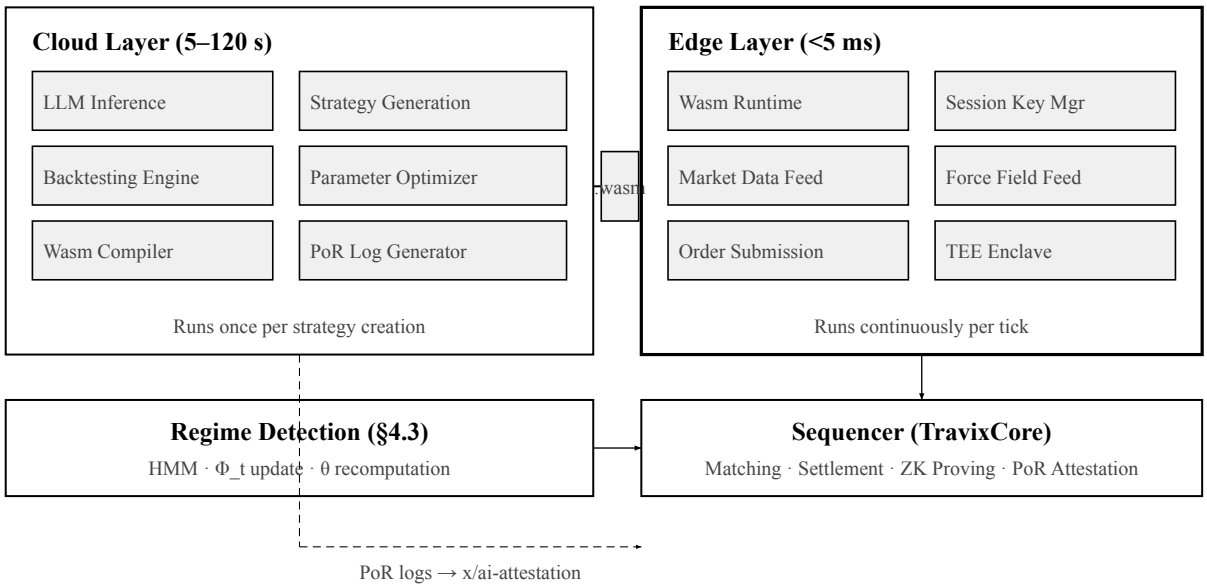

Figure 6: Cognitive-Execution Decoupled (CED) Architecture. The Cloud Layer generates strategies on a seconds-to-minutes timescale; the Edge Layer executes them at sub-5ms latency co-located with the Sequencer. Bold border denotes the trust boundary.

Delegation from the user's master key to the Edge Layer is mediated by Session Keys:

$$\text{SessionKey} = \text{Sign}(sk_{master}, \{allowed\_markets, max\_leverage, max\_order\_size, expiry\}) \tag{18}$$

The chain verifies constraint parameters before executing any order signed by a Session Key. Constraints are enforced *within the ZK circuit*: even a compromised Edge Node cannot exceed the user-specified leverage or position size limits. Keys are revocable at any time via a priority transaction that bypasses the normal queue, and the system enforces a Dead Man's Switch: if no heartbeat is received within a configurable timeout (default 60 seconds), all open orders are cancelled and positions are reduced to a governance-defined safe state.

Failover and Resilience. Each Edge Node maintains a heartbeat connection to the Cloud Layer. If the Cloud Layer becomes unreachable (LLM provider outage, network partition), the Edge Layer continues executing the last-deployed Wasm module with the last-known regime parameters—degraded but functional. The Wasm module's deterministic nature ensures that even without regime updates, execution remains predictable and auditable. Upon reconnection, the Cloud Layer pushes accumulated regime state changes, and parameters are updated atomically.

4.3. Regime-Adaptive Strategy Manifold

A fundamental limitation of current AI trading systems is that generated strategies use *static parameters*—fixed stop-loss percentages, fixed grid spacings, fixed position sizes—that fail when market regimes shift. When implied volatility doubles, a 2% stop-loss that was conservative becomes dangerously tight. We introduce a formal framework for dynamic parameter adaptation.

Definition 8 (Regime-Adaptive Strategy). A strategy S is decomposed as:

$$S = L(\theta(\Phi_t)) \tag{19}$$

where L is the strategy logic (static, compiled to Wasm), θ is the parameter vector, and Φ_t is the market regime state at time t. The logic L is generated by the LLM and remains fixed after compilation. The parameters θ are *functions of the regime state* Φ_t, updated in real time.

4.3.1. Market Regime State Space

The regime state $\Phi_t = (\sigma_{implied}, \sigma_{realized}, \text{funding\_rate}, \rho_{cross}, \text{VIX}, \text{DXY}, ...)$ is a vector of observable market features. Market conditions are classified into K discrete regimes via a Hidden Markov Model [10] with transition probability matrix:

$$P_{ij} = Pr(S_{t+1} = s_j \mid S_t = s_i) \tag{20}$$

Typical regime classification: s_1 = low-volatility trending, s_2 = high-volatility mean-reverting, s_3 = crisis/regime break. Detection signals include the realized-to-implied volatility ratio, funding rate magnitude and direction, and eigenvalue distribution of the cross-asset correlation matrix.

4.3.2. Adaptive Parameter Functions

Each strategy parameter θ_k is a function of the regime state:

$$\theta_k = f_k(\Phi_t) \tag{21}$$

Concrete mappings with financial interpretations:

$$\text{stop\_loss}(\sigma) = \text{base\_sl} \times (\sigma_{implied} / \sigma_{baseline}) \tag{22}$$

Higher volatility → wider stops, preventing premature exit during normal fluctuations.

$$\text{grid\_spacing}(\sigma) = \text{base\_grid} \times (\sigma_{implied} / \sigma_{baseline})^{\alpha} \tag{23}$$

Sparse grids in high volatility (avoid cascade fills), dense in low volatility (capture small moves).

$$\text{position\_size}(\text{VIX}) = \text{base\_size} \times exp(-\beta \times max(0, \text{VIX} - \text{VIX}_{th})) \tag{24}$$

Exponential position reduction as VIX exceeds fear threshold.

$$\text{entry\_threshold}(f) = \text{base\_th} + \gamma \times |\text{funding\_rate}| \tag{25}$$

Extreme funding rates imply crowded positioning; raise the bar for new entries.

The Wasm module contains both the static logic L and the parameter functions f_k. At each tick, the Edge Node injects the latest Φ_t into the Wasm runtime, which recomputes all θ_k before executing. Each parameter update (Φ_t, θ) is logged as an A5-class PoR attestation.

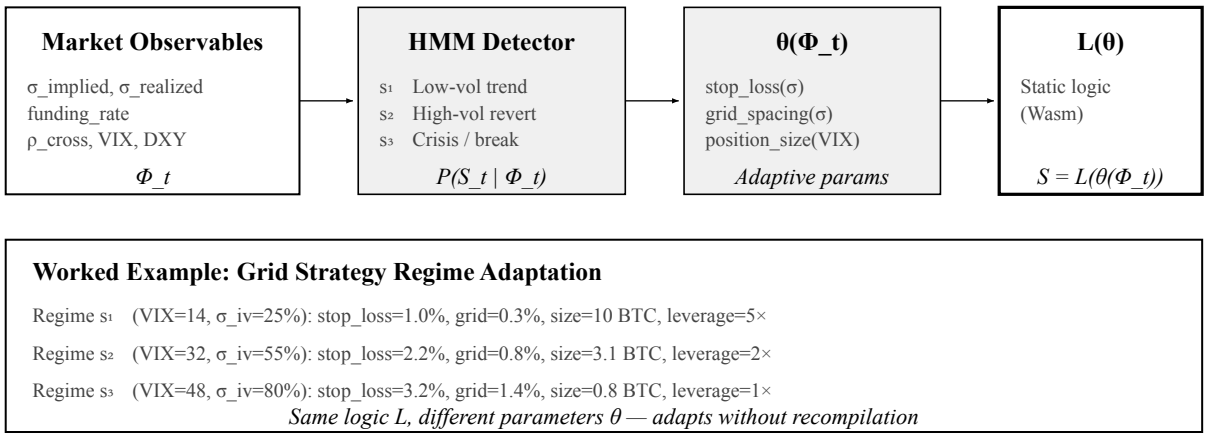

Figure 7: Regime-Adaptive Strategy Manifold. Market observables Φ_t flow through HMM regime detection, producing adaptive parameters θ that are injected into the static strategy logic L. The worked example shows concrete parameter values across three regimes for a grid trading strategy.

4.3.3. Regime Transition Smoothing

Abrupt parameter changes at regime boundaries can cause destabilizing behavior: a sudden halving of position size triggers a large market sell, which itself reinforces the high-volatility regime signal, creating a feedback loop. The system mitigates this through exponential smoothing of regime probabilities:

$$\theta_k(t) = \sum_{r=1}^{K} P(S_t{=}s_r) \cdot f_k(s_r) \tag{26}$$

The regime probability vector $P(S_t)$ is itself smoothed with an exponential moving average (half-life configurable, default 15 minutes), ensuring that parameters interpolate smoothly between regimes rather than jumping discretely. During the transition from s_1 to s_2, the strategy operates with blended parameters reflecting the weighted probability of each regime, reducing unnecessary position churn while still adapting directionally.

4.4. Trading Clone Engine

The Clone Engine generates unique AI trading agents from personal trading history, transforming implicit human intuition into explicit, executable strategy logic. The pipeline operates in five stages.

Stage 1: Data Authorization. Users authorize read-only access to historical trade data via a scoped API key. Critically, raw trade data never leaves the Travix infrastructure boundary: all feature extraction and LLM analysis occur within TEE enclaves, and the user may revoke access at any time. Only the final compiled Wasm module—which does not contain any raw data—is stored persistently.

Stage 2: Feature Extraction. The system extracts a high-dimensional feature vector $v_{\text{user}} \in \mathrm{R}^d$ capturing the trader's behavioral fingerprint across multiple dimensions: win rate by market condition, risk-reward ratio distribution, typical holding periods (seconds, minutes, hours, days), asset class

preferences, maximum observed drawdown tolerance, entry style (breakout vs. mean-reversion vs. momentum), exit patterns (trailing stop vs. fixed target vs. time-based), and correlation with external signals (funding rates, open interest changes, volume spikes). The feature space is designed to be rich enough that $||v_i - v_j|| > \varepsilon$ for any two distinct traders with meaningful history, ensuring strategy uniqueness.

Stage 3: Logic Extraction. An LLM analyzes the feature vector alongside raw trade sequences to identify core decision patterns: "tends to enter long after three consecutive red candles with declining volume," "always sets stop-loss at the nearest support level," "scales into positions in thirds." The LLM produces a structured strategy specification in an intermediate representation.

Stage 4: Constraint Application. Users specify natural language constraints that override or modify the extracted behavior: "never use more than 3× leverage" or "don't trade during Asian session." These constraints are compiled into hard-coded Wasm guards that cannot be overridden by the strategy logic.

Stage 5: Compilation and Deployment. The strategy specification is compiled into a deterministic Wasm module with regime-adaptive parameter functions (§4.3), backtested against historical data (§4.5), and deployed to the Edge Layer.

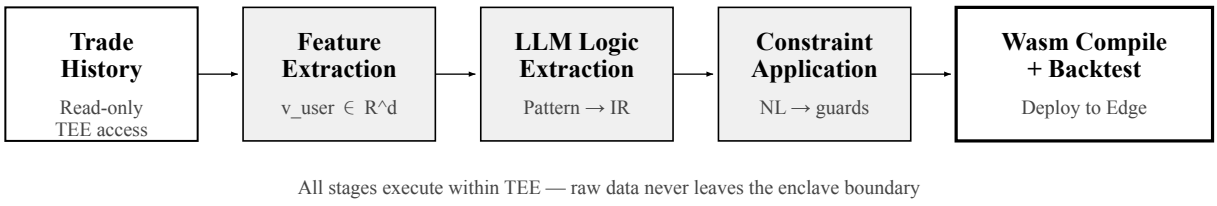

Figure 8: Clone Engine pipeline. Trade history is processed entirely within TEE enclaves, producing a unique Wasm strategy module. Bold border denotes the final deployable artifact.

4.5. Backtesting Engine

A columnar in-memory engine built on a Rust/Polars architecture supports tick-level replay over multi-year horizons with realistic friction modeling. The engine processes over 1 million candles in under 2 seconds on standard cloud compute, enabling rapid strategy iteration. Friction modeling includes: exchange fees (maker/taker with volume-tiered schedules), funding rate payments (applied at 8-hour intervals), slippage (modeled from historical order book depth at each timestamp), partial fill simulation (orders interact with reconstructed depth), and liquidation cascade effects (positions are closed progressively when margin thresholds are breached during replay).

Output metrics span risk-adjusted return measures (Sharpe, Sortino, Calmar ratios, CAGR), drawdown analysis (maximum drawdown, maximum drawdown duration, recovery time), trade-level statistics (win rate, profit factor, maximum consecutive losses, average holding period), and regime-con-

ditional breakdowns (performance decomposed by HMM regime state, revealing whether a strategy's returns are concentrated in a single regime or robust across all three).

Delivery is dual-track: human users receive interactive TradingView charts with annotated entry/exit markers, equity curves, and drawdown bands; AI agents receive structured JSON via MCP, enabling programmatic analysis and automated parameter iteration. An agent may autonomously execute grid search or Bayesian optimization over the parameter space, evaluating hundreds of parameter combinations in minutes and converging on Pareto-optimal configurations balancing return against drawdown.

4.6. Privacy-Preserving AI Execution

AI agents' strategy logic constitutes core intellectual property. When a Wasm module executes on an Edge Node, the node operator could inspect the bytecode. Travix addresses this through TEE-shielded execution: the Wasm runtime operates within a hardware enclave (SGX/TDX), with Remote Attestation proving to the user that execution occurs in a genuine enclave. End-to-end encryption is achieved: strategy logic decides within the TEE, generates a Pedersen-committed order directly within the same enclave, and submits the commitment to the Sequencer—at no point is the plaintext order or strategy visible outside the enclave.

PoR and privacy are balanced through opt-in transparency: attestation hashes commit to input-output pairs without revealing the actual data. Users choose which parties may access full logs for audit.

4.7. Strategy Marketplace and Compute Economics

The Strategy Marketplace is a curated registry where creators publish strategies as either open-source (full Wasm source available for inspection) or blackbox (encrypted Wasm, TEE-only execution) modules. A strict backtest/live separation prevents overfitting fraud: the system independently re-runs backtests on holdout data periods that the creator did not have access to during development, and publishes both creator-reported and independently-verified performance metrics.

Monetization. Three revenue models are supported: (1) subscription-based access (flat monthly USDC fee), (2) performance fees under a high-water mark model (creator receives a percentage of profits only when the strategy exceeds its previous peak equity), and (3) trading fee revenue share (creator receives a fraction of the trading fees generated by users running the strategy). Revenue is distributed on-chain via the `x/vault` module, ensuring transparent and auditable payments.

Governance and Curation. New strategies undergo a review process before listing: automated checks verify Wasm determinism, absence of external network calls, and compliance with resource limits (maximum memory, maximum execution time per tick). A reputation system tracks creator history: strategies that outperform their backtests accrue positive reputation; those that underperform or exhibit anomalous behavior (sudden parameter changes, concentration in illiquid markets) receive warnings. Strategies with sustained negative reputation are delisted.

Compute Economics. USDC balances serve dual purpose: trading margin and compute payment. Consumption—LLM tokens, backtest CPU-hours, Edge Node uptime, Oracle Terminal data feeds—is metered and settled daily from an off-chain Redis ledger reconciled against on-chain balances. A compute circuit breaker suspends agent execution when available balance drops below a configurable threshold, preventing agents from consuming margin intended for position maintenance. Users may set per-agent compute budgets independently of their trading margin.

5. Oracle Terminal: Probabilistic Event Trading Engine

5.1. The Probabilistic World Model

The Oracle Terminal's core insight is philosophical: *event trading is not trading prices; it is trading probabilities of future world states*. Every prediction market contract is an Arrow-Debreu security [11]: it pays $1 if and only if a specific state is realized. The contract price equals the risk-neutral probability of that state. When these probabilities shift but the corresponding asset prices have not yet responded, an informational arbitrage exists.

Concrete Example. Consider a Federal Reserve rate decision. At 13:45 EST, prediction market contracts for "Fed holds rates unchanged" shift from 72% to 58% over 15 minutes as informed participants incorporate leaked dot-plot expectations. Historical event studies show that a 14-percentage-point shift in this contract's probability corresponds to approximately a 0.8% move in DXY (negative), a 1.2% move in XAU (positive), and a 2.5% move in BTC (positive). Yet at 13:50 EST, BTC has moved only +0.3%—the remaining +2.2% of implied movement has not yet been priced in. The Oracle Terminal detects this discrepancy automatically: the KL-divergence surprise is high (probability moved from a relatively certain region), the transmission coefficient is statistically significant ($p < 0.01$ from historical calibration), and the information edge ΔT is positive (prediction market led asset by ~5 minutes). The system generates a long BTC signal with confidence proportional to the magnitude of the unexploited probability delta.

Definition 9 (Probability Delta). For event E with prediction market probability $P(E)$ and asset A with historical sensitivity $S(A,E)$:

$$\Delta = [P(E) \cdot S(A,E)] - \Delta_{actual}(A) \tag{27}$$

When $|\Delta|$ is significantly positive, the asset has not priced in the prediction market's information. The *Information Latency* $\Delta T = T_{\text{media}} - T_{\text{prediction}}$ measures time advantage.

5.2. Information-Theoretic Event Field

We introduce an original framework for quantifying the *information content* of prediction market probability changes and modeling how this information *propagates* to asset prices through a calibrated transmission network. Unlike approaches that model probability dynamics per se, we model the *surprise generated by probability changes* and its *temporal decay* as it propagates across the event-asset network.

5.2.1. Surprise Measure via KL Divergence

When event e_j's probability shifts from p_{old} to p_{new}, the information content of this change is measured by the Kullback-Leibler divergence [12] between the posterior and prior Bernoulli

distributions:

$$s_j = D_{KL}(q \,||\, p) = p_{new}\, ln(p_{new}/p_{old}) + (1-p_{new})\, ln((1-p_{new})/(1-p_{old})) \tag{28}$$

This is fundamentally more principled than using raw Δp as the signal metric. A shift from 50% to 55% (5 percentage points) and a shift from 95% to 100% (also 5 percentage points) have vastly different information content: the former is a mild update to an uncertain belief, while the latter is a near-certain resolution. KL divergence captures this asymmetry: s_j is large when extreme probabilities shift (high surprise) and small when uncertain probabilities fluctuate (low surprise).

The *directional surprise* preserves the sign:

$$\delta_j = \text{sign}(\Delta p_j) \cdot s_j \tag{29}$$

Comparison with Traditional Approaches. Traditional event study methodology [13] measures abnormal returns around event windows, but operates post-hoc on discrete events. Volatility-based approaches (GARCH, stochastic volatility models) capture uncertainty but not directional information content. The KL-divergence framework is distinct in three ways: (i) it operates in real time on continuously updating probability streams rather than discrete event announcements; (ii) it quantifies the *information content* of probability changes rather than the changes themselves, correctly weighting shifts near probability extremes; (iii) it naturally handles multiple simultaneous events through superposition (Eq. 31), whereas traditional approaches typically analyze events in isolation. The resulting force field is best understood as a continuous, multi-event extension of the classical event study framework, operating at machine speed rather than analyst speed.

5.2.2. Temporal Decay Kernel

Information does not propagate instantaneously from prediction markets to asset prices. The impact of a surprise at time t' on an asset at time $t > t'$ decays exponentially:

$$f_{ij}(t) = T_{ij} \cdot \delta_j(t') \cdot exp(-\lambda_{ij} \cdot (t - t')) \tag{30}$$

where T_{ij} is the transmission coefficient (historical impact weight of event j on asset i), λ_{ij} is the decay rate (fast-propagating events like rate decisions have small λ; slow-propagating events like demographic shifts have large λ), and $t-t'$ is the elapsed time since the surprise.

5.2.3. Net Force Field Vector

The *net force* on asset a_i at time t includes both first-order (linear) and second-order (interaction) terms. Define the temporally-decayed surprise $g_j(t) = \sum_{t'<t} \delta_j(t') \cdot \exp(-\lambda_{ij}(t-t'))$. The full force field is:

$$F_i(t) = \sum_j T_{ij} \cdot g_j(t) + \sum_{j<k} \Gamma_{ijk} \cdot g_j(t) \cdot g_k(t) \tag{31}$$

The first term captures independent event effects via the transmission coefficients T_{ij}. The second term captures *pairwise event interactions* via the interaction tensor $\Gamma_{ijk} \in \mathrm{R}^{m \times n \times n}$. When $\Gamma_{ijk} > 0$, events j and k are synergistic for asset i (their combined effect is super-additive: e.g., inflation sur-

prise + recession signal = stagflation, whose impact on equities far exceeds the sum of parts). When $\Gamma_{ijk} < 0$, the events are antagonistic (sub-additive). The linear-only model is the special case $\Gamma = 0$.

In matrix notation for the instantaneous snapshot: $\mathbf{F} = T \cdot \mathbf{g} + \mathbf{g}^{\mathrm{T}}\, \Gamma\, \mathbf{g}$, where the quadratic form captures all pairwise interactions. The interaction tensor is sparse in practice: most event pairs do not interact, and Γ_{ijk} is nonzero only for economically linked event pairs (calibration details in §5.2.5).

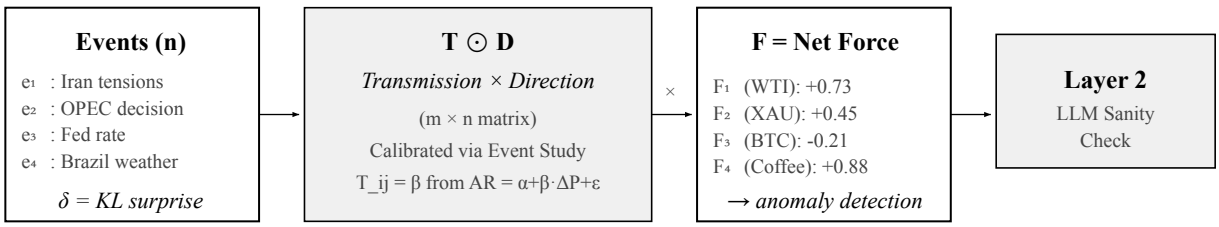

Figure 9: Information-Theoretic Event Field. KL-divergence surprises from prediction markets are propagated through the calibrated transmission matrix to produce asset-level net force vectors.

5.2.4. Anomaly Detection

The force field F_i is monitored against its historical distribution. The system triggers an escalation to Layer 2 when $|F_i - \mu_i| > 2\sigma_i$ where μ_i, σ_i are the rolling mean and standard deviation. Complementarily, the *mutual information* between event and asset is monitored:

$$I(E_j; A_i) = H(A_i) - H(A_i \mid E_j) \tag{32}$$

where H denotes Shannon entropy. When $\Delta I = I_{\text{current}} - I_{\text{baseline}}$ exceeds a threshold, the event is newly informative for the asset. The *Information Edge* metric combines surprise magnitude with time advantage: Edge $= s_j \times \Delta T$.

5.2.5. Transmission Coefficient Calibration

Transmission coefficients T_{ij} and interaction tensor entries Γ_{ijk} are calibrated via *regime-conditional* Event Study methodology [13]. For each HMM regime state r (§4.3.1), the abnormal return $\mathrm{AR}_{i,t} = R_{i,t} - \mathrm{E}[R_{i,t}]$ is regressed against probability changes and their pairwise interactions:

$$AR_i = \alpha_r + \sum_j T^r_{ij} \cdot \Delta P_j + \sum_{j<k} \Gamma^r_{ijk} \cdot \Delta P_j \cdot \Delta P_k + \varepsilon \tag{33}$$

The regime-specific coefficients T^{r}_{ij} capture the critical insight that the same event has different market impact across regimes: a 5% shift in Fed rate expectations during a low-volatility trend (s_1) produces a qualitatively different asset response than during a crisis regime (s_3). Interaction terms Γ^{r}_{ijk} are nonzero only for economically linked event pairs (e.g., inflation × employment, oil supply × geopolitical tension), identified via Lasso regularization to prevent overfitting.

Estimation uses *Huber M-estimators* rather than OLS to ensure robustness to outliers and black-swan observations that would otherwise dominate parameter estimates. The estimation window uses expanding-window cross-validation with regime-stratified sampling. Decay rates λ_{ij} are calibrated from the half-life of cumulative abnormal returns following historical event probability shifts. When the force field magnitude $|F_i|$ exceeds $3\sigma_i$—indicating an event of historically unprecedented magnitude—the system switches from the calibrated T_{ij} to a *crisis prior*: a conservative transmission estimate derived from a broader class of historical crises, acknowledging that out-of-sample events invalidate in-sample parameter estimates. The resulting T_{ij} with p-value and R^2 feeds into Signal Scoring dimension D_3.

5.2.6. Bayesian Cross-Event Dependencies

Events are not independent: Iran tensions correlate with oil supply fears, which correlate with inflation expectations. The system models cross-event dependencies through formal Bayesian conditional probability updates. When event e_a's probability shifts by ΔP_a, the posterior probability of a conditionally dependent event e_b is updated via:

$$P(e_b \mid \Delta e_a) = P(e_b) \cdot L(\Delta e_a \mid e_b) / Z \tag{33'}$$

where $L(\Delta e_a \mid e_b)$ is the likelihood of observing event a's probability shift given event b's state, calibrated from historical conditional co-occurrence data using kernel density estimation. The normalizer Z ensures the posterior is a valid distribution. This Bayesian update cascades: the revised $P(e_b)$ generates a secondary surprise δ_b', which feeds back into the force field (Eq. 31) as a second-order effect—precisely the mechanism captured by the interaction tensor Γ_{ijk}. The cascade depth is bounded at two hops to prevent runaway feedback loops.

5.2.7. Probability Matrix and Causal Network

The complete force field state is presented as a *Probability Matrix*: rows are active events (sorted by $|\delta_j|$), columns are tradable assets, cells contain (direction, T_{ij}, expected magnitude, time window). The bottom row aggregates the net force. A *Causal Event Network* visualizes the same data as a force-directed graph: event nodes connected to asset nodes by edges whose thickness encodes T_{ij} and whose color encodes direction. Users may perform What-If simulations by dragging probability sliders, triggering real-time force field recomputation.

Each force field computation generates an A1-class PoR attestation: H(events ‖ T ‖ $\mathbf{F}$) is committed to the Attestation Merkle Tree.

5.3. Three-Layer Decision Architecture

The Oracle Terminal processes information through three layers of increasing depth and decreasing speed, each layer adding context that the previous layer cannot provide. Escalation between layers is

automatic: anomalous force field readings trigger LLM analysis, and high-impact events trigger full swarm simulation.

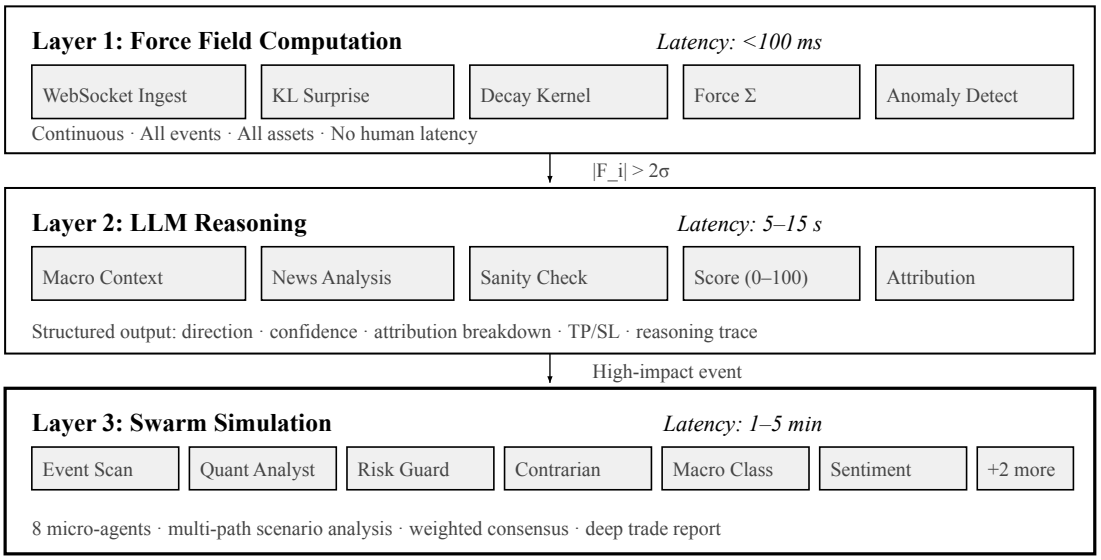

Figure 10: Three-Layer Decision Architecture. Each layer adds depth at the cost of latency. Escalation is automatic based on anomaly magnitude and event importance. Bold border denotes the deepest analysis layer.

Layer 1 — Force Field Computation (millisecond). WebSocket streams from prediction market APIs (Polymarket, Kalshi, PredictIt, and decentralized alternatives) feed continuous force field updates. The system ingests probability changes, computes KL-divergence surprises (§5.2.1), applies temporal decay kernels (§5.2.2), and produces updated net force vectors (§5.2.3) for all monitored assets. This layer operates continuously and autonomously, requiring no human intervention. Anomalous forces ($|F_i| > 2\sigma_i$) trigger automatic escalation to Layer 2.

Layer 2 — LLM Reasoning (second). When escalated, the LLM receives the anomalous force field vector alongside macro context (current VIX level, DXY trend, yield curve shape, recent central bank communications) and real-time news headlines. Its task is to perform a "sanity check" on the mathematical signal: does the force field's directional implication make fundamental sense given the broader context? Output is a structured assessment containing: direction (long/short/neutral), confidence score (0–100), attribution breakdown (percentage contribution of each contributing event to the net force), suggested take-profit and stop-loss levels, and a natural language reasoning trace. Attribution is versioned over time: if the LLM's reasoning for the same signal diverges significantly between consecutive invocations, the system flags potential instability and reduces position sizing. Users may interactively challenge the AI with counterarguments ("What about the upcoming CPI print?"), forcing re-analysis with the additional context incorporated.

Layer 3 — Swarm Simulation (minute). For major events with potentially outsized market impact—elections, central bank decisions, geopolitical escalations, regulatory announcements—the system spawns eight specialized micro-agents that conduct independent multi-path scenario analysis. Each agent contributes a distinct analytical perspective: the *Event Scanner* identifies all potentially corre-

lated events and their current probability states; the *Quantitative Analyst* runs Monte Carlo simulations of asset price paths under each scenario; the *Risk Guardian* identifies worst-case tail scenarios and maximum portfolio exposure; the *Contrarian Thinker* deliberately argues against the consensus direction to stress-test the thesis; the *Macro Regime Classifier* assesses whether the event could trigger a regime transition (§4.3.1); the *Sentiment Analyst* evaluates social media and news sentiment for confirmation or divergence; the *Liquidity Assessor* checks order book depth and estimates execution impact; and the *Portfolio Optimizer* proposes optimal position sizing across the correlated asset set. Agent outputs are aggregated via a weighted consensus mechanism, with weights derived from each agent's historical accuracy on similar event types. The final output is a deep trade report containing probability distributions across outcomes, recommended positions, and scenario-conditional risk parameters.

5.4. Signal Scoring Framework

$$CS = 0.30 \cdot D_1 + 0.20 \cdot D_2 + 0.20 \cdot D_3 + 0.15 \cdot D_4 + 0.15 \cdot D_5 \tag{34}$$

where D_1 = Probability Delta magnitude ($|\Delta|$ from Eq. 27; veto if $\Delta \approx 0$), D_2 = Velocity dp/dt (faster shifts carry higher information value), D_3 = Historical Reliability (p-value and R^2 from Event Studies), D_4 = Force Field Consistency (cosine similarity of contributing event force vectors: when all events push in the same direction, $\cos\theta \to 1$; when they cancel, $\cos\theta \to 0$), D_5 = Macro Regime Context (alignment with the current Hamilton regime state from §4.3.1). The score admits a Bayesian interpretation: $CS \approx 100 \cdot \sigma(w^{\mathrm{T}}x)$, with weights calibrated via logistic regression on historical signal-outcome pairs. Breakdowns are committed as A3-class attestations.

5.5. Event-Conditional Order System

The matching engine supports event-triggered orders stored in the Instruction Stack. Types include: single-event threshold (IF $P(E_x) >$ threshold THEN execute), compound conditions (IF $P(A) > 60\%$ AND $P(B) > 40\%$ THEN ...), velocity triggers (IF $dp/dt > 15\%$/hour THEN ...), and force field triggers (IF $|F_i| > 2\sigma$ THEN ...). When conditions are met, the Force Field Engine converts them to standard orders entering the TravixOBT matching flow.

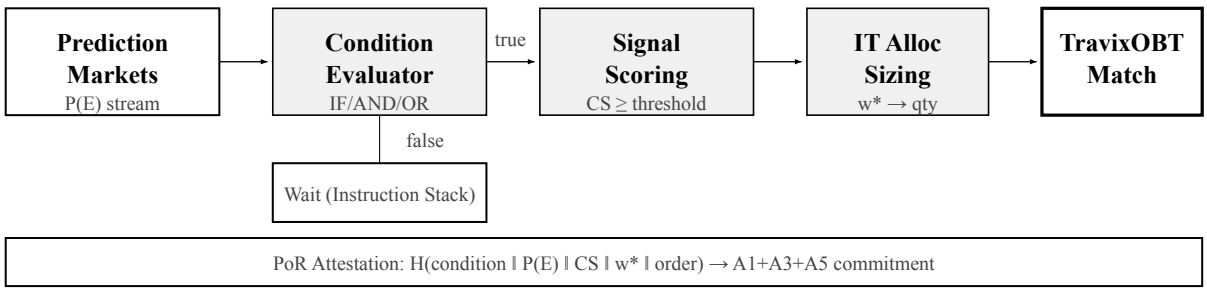

Figure 11: Event-Conditional Order lifecycle. Probability stream feeds condition evaluation; satisfied conditions flow through signal scoring and information-theoretic allocation before entering the TravixOBT matching engine. All stages generate PoR attestations.

5.6. Information-Theoretic Position Allocation

A principled position sizing framework should derive naturally from the same information-theoretic foundation that generates the signal itself. We formalize the allocation problem as minimizing the divergence between the portfolio's event exposure and the force field's directional signal, subject to risk constraints.

Definition 10 (Optimal Event-Aligned Allocation). Let $\mathbf{w} = (w_1, ..., w_m)$ be portfolio weights across m assets, and $\mathbf{F}_{\text{norm}} = \mathbf{F} / ||\mathbf{F}||_1$ be the L1-normalized force field vector. The optimal allocation minimizes the information divergence between the portfolio's implied event exposure and the force field signal:

$$\boldsymbol{w}^* = \text{argmin}_{\boldsymbol{w}} \, D_{KL}(\text{Exposure}(\boldsymbol{w}) \parallel \boldsymbol{F}_{norm}) \tag{35}$$

subject to: $\sum|w_i| \leq L_{\max}$ (leverage constraint), $\text{CVaR}_\alpha(\mathbf{w}) \leq c$ (tail risk budget), and $w_i \in [-w_{\text{cap}}, w_{\text{cap}}]$ (per-asset concentration limit).

Here $\text{Exposure}(\mathbf{w}) = T^{\text{T}} \cdot \mathbf{w}$ maps portfolio weights through the transposed transmission matrix to produce an event-space exposure vector. The optimization ensures that capital is allocated proportionally to the information content arriving from each event channel, weighted by the transmission coefficient's statistical reliability.

The conditional Value-at-Risk constraint $\text{CVaR}_\alpha(\mathbf{w})$ incorporates scenario stress testing from §5.8: for each pending event e_k, the system computes E[PnL | e_k=Yes] and E[PnL | e_k=No], ensuring that neither scenario triggers liquidation. The leverage bound $L_{\max}$ is itself a function of the force field magnitude: higher aggregate $|\mathbf{F}|$ (stronger, more consistent signals across events) permits higher leverage, while a fragmented or contradictory force field automatically constrains exposure. This creates a self-regulating system where position sizing is mathematically coupled to signal quality through a unified information-theoretic framework.

5.7. Lifecycle Exit Mechanism

Every Oracle Terminal position has a finite lifecycle governed by the information edge that justified its entry. Four conditions trigger automatic exit (any one is sufficient):

(1) Convergence. The probability delta that generated the signal has been priced in: $|\Delta| < \varepsilon$ where ε is a market-specific threshold (default: 0.05× the entry delta). The asset price has moved to reflect the prediction market's information, eliminating the informational arbitrage. Take-profit is typically set at the convergence target implied by the entry signal.

(2) Reversal. The force field vector F_i reverses sign (the directional thesis is invalidated by new probability updates), or the position's unrealized loss exceeds a hard stop (default: 2% of position notional, adjustable by regime state per §4.3). When reversal is triggered by a force field sign flip, the system additionally checks whether the reversal represents a genuine change in information or merely noise, using the same anomaly detection threshold (2σ) as the entry signal.

(3) Invalidation. The prediction market contract resolves to NO, is disputed, or the event is cancelled. In these cases, the information source that generated the signal no longer exists, and the position is closed immediately regardless of PnL. The system monitors contract resolution status via WebSocket feeds and triggers closure within one tick of detection.

(4) Time Decay. The position exceeds the expected impact horizon without significant price movement in the predicted direction. The impact horizon is estimated from the calibrated decay rate λ_{ij} (§5.2.2): if more than $3/\lambda_{ij}$ time has elapsed (approximately three half-lives of the expected information propagation), the signal is considered exhausted. Remaining positions are reduced proportionally over a TWAP window to minimize market impact.

5.8. Event-Driven Portfolio Risk

Event Exposure Map. The portfolio's exposure to event j:

$$\mathrm{EE}_j = \sum_i weight_i \times T_{ij} \tag{36}$$

Concentration risk = $\max|\mathrm{EE_j}| \,/\, \sum|\mathrm{EE_j}|$. The system warns when a seemingly diversified portfolio is hidden-concentrated on a single event.

Scenario Stress Testing. For pending event e_k, the system computes E[PnL | e_k=Yes] and E[PnL | e_k=No], including whether either scenario triggers liquidation. Auto-deleveraging reduces positions proportionally or closes lowest-confidence trades when margin ratio approaches danger levels.

5.9. Implementation Considerations

Data Source Reliability. The Oracle Terminal's accuracy depends critically on the quality and reliability of prediction market probability feeds. The system addresses this through multi-source aggregation: probabilities for the same event are ingested from multiple prediction market platforms (where available) and combined via inverse-variance weighting, reducing the impact of any single platform's noise or manipulation. When a platform's price deviates from the weighted consensus by more than 3σ, it is flagged as potentially manipulated and down-weighted. For events covered by only a single platform, the system applies a reliability discount to the confidence score (dimension D_3 in Eq. 34).

Latency Requirements. The information edge exploited by the Oracle Terminal has a finite half-life: once prediction market probabilities shift, the corresponding asset price adjustment typically begins within 30 seconds to 15 minutes depending on the event type and asset liquidity. The system's end-to-end latency from probability change detection to order submission must remain well below this

propagation timescale. Current architecture achieves sub-200ms from WebSocket receipt to force field update (Layer 1), with order submission adding sub-5ms when using the Edge Layer. The latency budget is dominated by LLM inference (Layer 2), which is why Layer 1 can operate autonomously for routine signals, reserving LLM analysis for anomalous conditions.

Fallback Mechanisms. In the event of prediction market API failures (platform outages, rate limiting, network partitions), the Oracle Terminal degrades gracefully: existing positions continue to be managed using the last-known force field with accelerated time decay (the decay rate λ is doubled during data outages, reflecting increased uncertainty). No new event-conditional orders are activated during outages. When connectivity is restored, the system performs a catch-up reconciliation, computing the cumulative surprise over the outage period and adjusting the force field accordingly.

6. Conclusion

This paper has presented the Travix Protocol, addressing four structural deficiencies in decentralized derivatives: asset isolation, tool asymmetry, information latency, and AI opacity. The TravixOBT data structure advances verifiable order book design with margin-aware aggregate nodes achieving $\Theta(\log N)$ for all operations including margin verification—a complexity improvement from $O(N)$ that is, to our knowledge, novel in the verifiable order book literature. The Regime-Adaptive Strategy Manifold introduces a formal separation $S = L(\theta(\Phi_t))$ of static strategy logic from dynamic market-regime-conditioned parameters, with HMM-based regime detection and smooth parameter interpolation providing the mathematical foundation for AI agents whose behavior adapts continuously to market conditions without the catastrophic failures characteristic of static parameter systems. The Information-Theoretic Event Field establishes an original framework for quantifying prediction market information content via KL-divergence surprises propagated through calibrated transmission networks with exponential temporal decay—transforming the force field from a heuristic concept into a principled, real-time signal-processing system with formal anomaly detection and cross-event dependency modeling. The Proof of Reasoning protocol provides, to our knowledge, the first systematic approach to on-chain AI decision auditability in a production trading system, co-verified within the same Groth16 proof that attests to trading correctness.

Limitations and Future Work. Several limitations warrant acknowledgment. First, the Proof of Reasoning provides auditability (the AI used the data it claims) but not verifiability (the AI computed correctly); full zkML integration [8] remains a future milestone. Second, the Oracle Terminal's effectiveness depends on prediction market liquidity and accuracy, which vary significantly across event types and geographies. Third, the regime-adaptive framework assumes that market dynamics can be adequately captured by a finite number of discrete regimes; continuous regime spaces and regime-agnostic adaptive methods remain areas for exploration. Fourth, the TEE is the weakest link in the privacy model: SGX has been subject to multiple side-channel attacks [20], and while the protocol ensures that TEE compromise degrades only privacy (not correctness), the information leakage within a batch window provides MEV-exploitable advantage. Migrating to hardware-independent privacy primitives (fully homomorphic encryption for matching, or MPC-based dark pools) remains an ac-

tive research direction. Fifth, the force field's interaction tensor Γ_{ijk} introduces $O(n^2)$ calibration parameters for n events; regularization (Lasso) and sparsity priors mitigate overfitting, but novel event combinations outside the training distribution remain a fundamental challenge for any data-driven model. Future work will address these limitations through progressive decentralization of the prover infrastructure, expansion of the zkML verification frontier, and development of formal game-theoretic models for the Oracle Terminal's incentive structure.

Together, these contributions position Travix not as another perpetual futures exchange, but as a new category of financial infrastructure: the verifiable, AI-augmented, event-driven omnibroker.

References

[1] Hyperliquid Foundation. "Hyperliquid: A High-Performance L1 for On-Chain Finance." 2024.

[2] dYdX Foundation. "dYdX v4: Decentralized Perpetual Exchange Protocol." 2023.

[3] Elliot Technologies (Lighter). "Lighter Protocol: Order Book Matching and Liquidations with Transparent and Verifiable Computation." October 2025.

[4] Polymarket. "Polymarket: Information Markets Protocol." 2024.

[5] Buchman, E. "Tendermint: Byzantine Fault Tolerance in the Age of Blockchains." Ph.D. Thesis, 2018.

[6] Grassi, L. et al. "Poseidon: A New Hash Function for Zero-Knowledge Proof Systems." *USENIX Security*, 2021.

[7] Pedersen, T. P. "Non-interactive and information-theoretic secure verifiable secret sharing." *CRYPTO '91*.

[8] EZKL. "EZKL: Easy Zero-Knowledge Machine Learning." 2024.

[9] Anthropic. "Model Context Protocol." 2024.

[10] Hamilton, J. D. "A New Approach to the Economic Analysis of Nonstationary Time Series and the Business Cycle." *Econometrica*, 57(2), 1989.

[11] Arrow, K. J. "The Role of Securities in the Optimal Allocation of Risk-bearing." *Review of Economic Studies*, 31(2), 1964.

[12] Kullback, S. and Leibler, R. A. "On Information and Sufficiency." *Annals of Mathematical Statistics*, 22(1), 1951.

[13] MacKinlay, A. C. "Event Studies in Economics and Finance." *Journal of Economic Literature*, 35(1), 1997.

[14] Kelly, J. L. "A New Interpretation of Information Rate." *Bell System Technical Journal*, 1956.

[15] Groth, J. "On the size of pairing-based non-interactive arguments." *EUROCRYPT*, 2016.

[16] EdgeX DAO. "EDGE Stack: App-Specific Execution Layer for On-Chain Derivatives." 2025.

[17] Daian, P. et al. "Flash Boys 2.0: Frontrunning in Decentralized Exchanges." *IEEE S&P*, 2020.

[18] Amdahl, G. M. "Validity of the single processor approach." *AFIPS*, 1967.

[19] Hanson, R. "Combinatorial Information Market Design." *Information Systems Frontiers*, 2003.

[20] Costan, V. and Devadas, S. "Intel SGX Explained." IACR ePrint 2016/086.

[21] Cosmos Network. "Cosmos SDK Documentation." cosmos.network, 2019.

[22] Nakamoto, S. "Bitcoin: A Peer-to-Peer Electronic Cash System." 2008.