EDGE STACK: THE APP-SPECIFIC EXECUTION LAYER FOR HIGH-FREQUENCY ON-CHAIN DERIVATIVES

EDGEX DAO INC

ABSTRACT

The sequential execution model of monolithic blockchains imposes a fundamental latency floor on decentralized derivatives, forcing asset-specific trading logic to compete for global state access. This paper introduces EDGE Stack, an App-Specific Execution Layer engineered for high-frequency on-chain derivatives trading. Unlike general-purpose rollups [9], EDGE Stack implements a Deterministic Parallel Transaction Execution (PTE) engine that leverages market-sharded state access to execute non-conflicting order books concurrently. Furthermore, EDGE Stack integrates a modular Multi-VM architecture for resource isolation alongside FlashLane, a dual-tiered transaction prioritization mechanism that decouples matching latency from asynchronous settlement. To ensure the verification of this parallelized state, the system employs Parallel Merklization, allowing state tree hashes to be computed concurrently and converged into a deterministic global state root. The resulting architecture combines the linear scalability and sub-second soft finality of centralized engines with the verifiable, cryptographic assurances of decentralized networks, currently leveraging Ethereum's security guarantees.

***K*eywords** Deterministic Parallel Execution · Parallel Merklization · Modular Multi-VM · Extended Access Lists · State Isolation

1 Introduction

1.1 The Infrastructure Gap in Decentralized Derivatives

Decentralized Finance (DeFi) has fundamentally reshaped market access, yet its most complex sector—perpetual futures and derivatives—remains severely constrained by underlying infrastructure. While Automated Market Makers (AMMs) revolutionized spot trading, they are inherently capital-inefficient for the leveraged, high-frequency trading capabilities required by mature derivatives markets [23]. Institutional-grade derivatives demand Central Limit Order Books (CLOBs) for precise price discovery and deep liquidity[11].

However, deploying high-performance CLOBs on current blockchain infrastructure presents a structural contradiction. Monolithic blockchains [8] utilize a sequential execution model over a single global state machine. In this environment, high-frequency trading logic must compete for scarce block space and execution resources against every other network activity. This "noisy neighbor" problem imposes an unpredictable latency floor and throughput ceiling, rendering general-purpose chains incapable of supporting the sub-millisecond speed and deterministic reliability demanded by professional trading firms.

The industry faces a critical bottleneck: **the demand for decentralized derivatives has vastly outpaced the capabilities of generalized execution environments.**

1.2 EDGE Stack: The App-Specific Execution Layer

EDGE Stack is engineered as the definitive solution to this infrastructure gap. It is a specialized App-Specific Execution Layer designed from the ground up with a singular mandate: to deliver the performance of a centralized trading venue with the cryptographic verifiability of a decentralized network.

By fundamentally decoupling high-performance execution from base-layer settlement, EDGE Stack moves beyond the constraints of general-purpose rollups. It provides a sovereign environment optimized exclusively for financial throughput, ensuring that trading activity is never hindered by unrelated network congestion. Ultimately, EDGE Stack provides the purpose-built, high-performance infrastructure foundational to the next generation of on-chain perpetual and derivatives markets.

1.3 Core Architectural Pillars

To resolve the conflict between high-frequency performance and decentralized trust, EDGE Stack introduces a novel architecture built upon three proprietary technical pillars optimized for the trading lifecycle.

- **Modular Multi-VM Architecture: Achieving Resource Isolation.** Unlike monolithic systems where all transactions share a single runtime, EDGE Stack implements a bifurcated environment that physically isolates high-performance trading logic from generalized smart contract execution. This architectural firewall ensures dedicated computational bandwidth for the critical matching engine, eliminating resource contention from "noisy neighbors" and guaranteeing consistent performance regardless of wider network loads. (Detailed in Section 3).
- **Deterministic PTE Engine: Unlocking Linear Scalability.** Moving beyond the throughput limits of sequential processing [21], EDGE Stack employs a deterministic parallelization engine. By leveraging the natural state isolation inherent across different derivative markets, the engine executes non-conflicting order books concurrently. Consequently, system throughput scales linearly with available hardware resources, achieving ultra-high speeds without succumbing to the non-determinism typical of optimistic parallel models. (Detailed in Section 4).
- **FlashLane: Delivering Institutional-Grade Latency.** To overcome the latency limitations of standard blockchain mempools, EDGE Stack introduces FlashLane, a dual-tiered protocol-native prioritization mechanism. By actively differentiating trading directives from asynchronous operations, FlashLane enables traders to receive instant, cryptographically assured soft-confirmations. This effectively decouples trade execution speed from the slower finality of the underlying base layer. (Detailed in Section 5).

1.4 State Convergence and Security Foundation

Achieving ultra-high execution throughput is futile if the underlying state commitment cannot keep pace. In traditional systems, the serial process of recomputing cryptographic state hashes (Merklization) becomes the primary bottleneck under high loads, introducing unpredictable latency jitter that is unacceptable for institutional trading.

To address this, EDGE Stack incorporates an advanced state management layer designed to operate concurrently with its execution engine. The system utilizes Parallel Merklization to compute state changes across isolated market shards simultaneously. This mechanism ensures that cryptographic consistency aligns precisely with matching engine speed, guaranteeing predictable, deterministic latency for state commitments while maintaining full on-chain verifiability without compromising top-tier performance. (Detailed in Section 6).

1.5 Conclusion

In conclusion, EDGE Stack resolves the foundational conflict between high-frequency performance and decentralized trust. By establishing a new paradigm rooted in App-Specific Execution, this architecture not only overcomes immediate scalability bottlenecks for derivatives but secures a resilient, future-proof foundation for global on-chain market infrastructure.

2 Architecture Overview

2.1 Design Philosophy: The App-Specific Execution Paradigm

The architectural foundation of EDGE Stack is predicated on the Modular Blockchain Thesis: that high-frequency finance demands a specialized execution layer fundamentally distinct from general-purpose consensus layers.

Monolithic chains force high-stakes derivative markets to compete for shared block space resources against unrelated traffic, leading to inevitable latency spikes and MEV leakage [17]. In contrast, EDGE Stack is engineered as a Sovereign Execution Environment designed exclusively for financial throughput.

Our design philosophy prioritizes full-stack specialization over generic optimization. By customizing the entire transaction lifecycle, ranging from the Sequencer's pre-confirmation logic down to specific Virtual Machine opcodes, we eliminate the computational overhead inherent in generalized state machines. In this paradigm, trading functions cease to be mere smart contracts and evolve into first-class protocol primitives. This decoupling of Execution from Settlement enables deterministic, ultra-low latency performance while inheriting the trust-minimized security properties of the underlying decentralized network.

2.2 The Stack Hierarchy

EDGE Stack operates through a layered topology designed for strict separation of concerns, where data flows through a distinct three-tier architecture ranging from user ingress to final settlement.

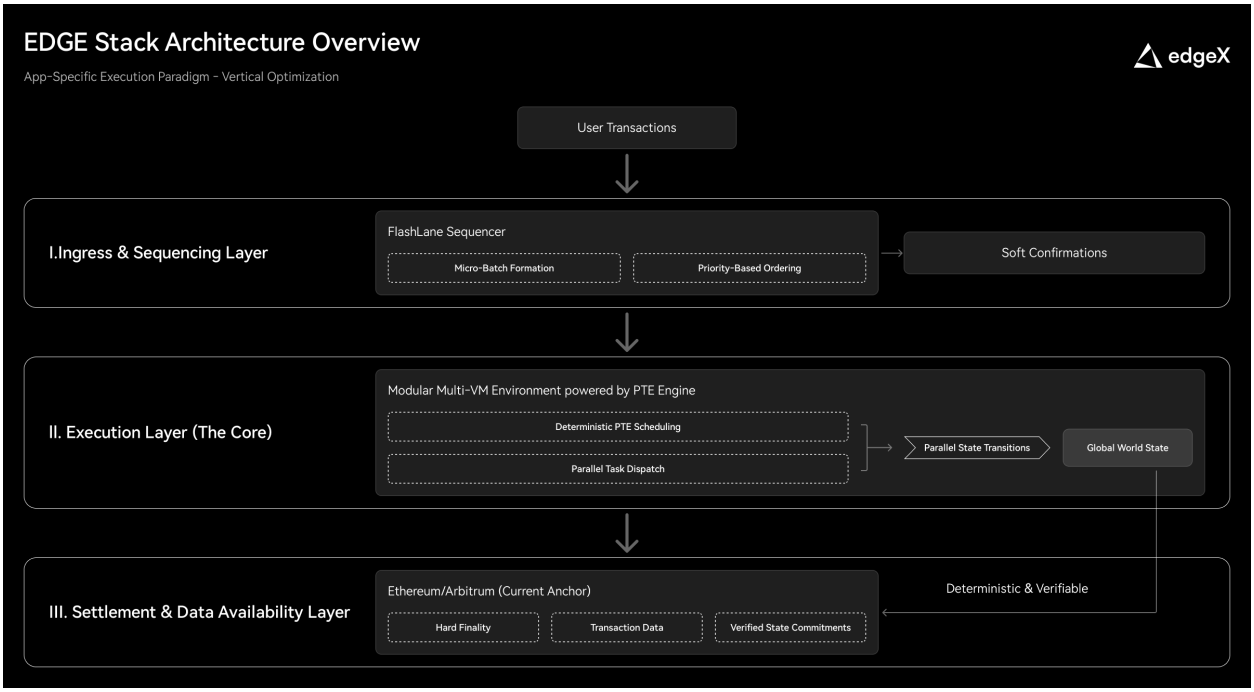

Figure 1: **The EDGE Stack Layered Architecture.**

2.2.1 I. The Ingress & Sequencing Layer

At the top of the stack sits the FlashLane Sequencer, acting as the primary gateway for transaction ingress. Its function goes beyond simple mempool aggregation; it actively performs traffic shaping and ordering based on operational urgency. By organizing chaotic network input into structured batch data, the Sequencer acts as a high-performance buffer that provides immediate deterministic soft-confirmations to clients prior to execution.

2.2.2 II. The Execution Layer (The Core)

At the heart of the stack lies the Execution Layer, the proprietary engine responsible for deterministic state transitions. This core integrates the **Modular Multi-VM** environment powered by the **PTE Engine** to transcend the throughput limitations of traditional serial execution models. Upon receiving ordered transaction batches from the Sequencer, the Execution Layer dynamically routes workloads into isolated, purpose-built runtime environments. Latency-critical trading logic is directed to the specialized edgeVM, while standard smart contract interactions are handled by the edgeEVM. Ultimately, the output of this parallel processing is not merely a set of disjoint state changes, but a series of parallel transitions cryptographically converged into a singular, verifiable Merkle Root ready for settlement.

2.2.3 III. The Modular Settlement & Data Availability Layer

The foundational layer handles crypto-economic security, immutability, and data availability (DA) [10]. Crucially, EDGE Stack treats this layer not as a fixed dependency but as a pluggable, modular interface. Its primary role is to serve

as the anchor for finality, where compressed transaction data and proven state roots are posted for trustless verification. While the initial implementation leverages established networks such as the Ethereum ecosystem for robust security assurances, the architecture remains agnostic to the specific underlying consensus mechanism. This modular design preserves strategic flexibility, allowing EDGE Stack to integrate with various DA solutions or evolve into a sovereign settlement network in the future.

2.3 Key Differentiator: Verifiable, Deterministic Parallelism

The defining characteristic that distinguishes EDGE Stack is its ability to reconcile high-throughput multi-module parallelism with strict cryptographic verifiability.

In conventional parallel architectures, achieving velocity often necessitates compromising determinism, introducing race conditions or hardware-specific timing variances that make trustless auditing impossible. EDGE Stack eliminates this compromise. By utilizing a PTE Engine and Parallel Merklization algorithm, the architecture guarantees that given an ordered input batch, parallel execution will always converge to the exact same State Root, regardless of the underlying hardware infrastructure.

Crucially, this strict adherence to determinism is maintained even under the extreme optimization pressures of **FlashLane's** instant pre-execution mechanisms. This capability is foundational to the Web3 ethos of **"Don't Trust, Verify,"** bridging the gap between the low-latency performance of centralized engines and the trustless auditability inherent to public blockchains [18].

3 Pillar I: Modular Multi-VM Architecture

3.1 The Limits of Monolithic Execution

General-purpose blockchains, designed for maximum versatility, become a liability for specialized financial applications. In a monolithic EVM architecture, heterogeneous tasks—from complex derivative pricing to simple governance votes—compete for identical resources under uniform metering logic [2]. This inherent resource contention imposes a structural ceiling on the performance of computationally intensive trading systems.

To overcome this bottleneck, EDGE Stack adopts a Modular Multi-VM Architecture based on the principle of Separation of Concerns [19]. By architecturally decoupling high-frequency trading logic from generalized smart contract execution, the system optimizes distinct runtime environments for their specific workloads while maintaining seamless interoperability.

3.2 Core Components: The Parallel Runtime Environment

The EDGE Stack execution layer implements a bifurcated runtime architecture, operating two primary environments in parallel alongside specialized extension mechanisms. This design ensures that highly specialized trading workloads are logically and physically segregated from generalized smart contract logic, optimizing each for its distinct operational profile.

3.2.1 I. edgeVM: The High-Performance Trading Runtime

The edgeVM is a minimalist execution environment engineered exclusively for high-frequency financial tasks, including the matching engine, real-time risk management, and liquidation logic.

- **Near-Native Execution:** By eschewing general-purpose computation, edgeVM eliminates the significant overhead inherent in the standard stack-based EVM architecture, such as instruction decoding and complex metering. Instead, trading logic is compiled into highly efficient WebAssembly (WASM) (leveraging advanced stacks such as Arbitrum Stylus [1]), allowing performance-critical code written in languages like Rust or C++ to achieve speeds rivaling bare-metal execution.
- **Dedicated Computational Bandwidth:** edgeVM operates within a protected execution context with guaranteed CPU and memory allocation. This architectural firewall ensures immunity to "noisy neighbor" congestion from generalized DeFi traffic, guaranteeing deterministic latency for critical trade execution regardless of wider computational load.

3.2.2 II. edgeEVM: The Standard Compatibility Runtime

Running in parallel with the trading core is the edgeEVM, a fully conforming Ethereum Virtual Machine designed to maintain robust interoperability with the broader ecosystem.

- **Standard DeFi Logic:** edgeEVM handles tasks that require EVM equivalence [20], such as ERC-20/721 asset issuance, DAO governance mechanisms, and treasury management.
- **Developer Experience:** This layer ensures seamless composability, allowing existing Solidity smart contracts and standard industry tooling (e.g., Hardhat, Foundry) to be deployed and utilized without modification.

3.2.3 III. VM Actors: The Modular Extensibility Framework

Beyond the primary bifurcated runtime for core trading and compatibility, the EDGE Stack architecture includes a dedicated framework for secure, future-proof extensibility. This framework is built on the "VM Actor" model, designed to resolve the tension between rapid innovation and systemic stability.

- **Future-Proof Scalability:** It allows the protocol to evolve horizontally by adding entirely new financial primitives—such as options exchanges, prediction markets, or experimental yield strategies—without requiring monolithic upgrades or network forks.
- **Hot-Swappable Modules:** New product lines are treated as independent, modular actors that can be dynamically deployed and integrated. This transforms the protocol from a static codebase into a flexible platform that can adapt to emerging DeFi trends and asset classes.
- **Architectural Safety:** Crucially, this framework is designed from the ground up to ensure that the addition of new, potentially experimental logic cannot compromise the security or performance integrity of the core perpetual trading engine. It provides the guardrails necessary for permissionless innovation on top of a robust financial infrastructure.

3.3 Deterministic Inter-VM Interoperability Framework

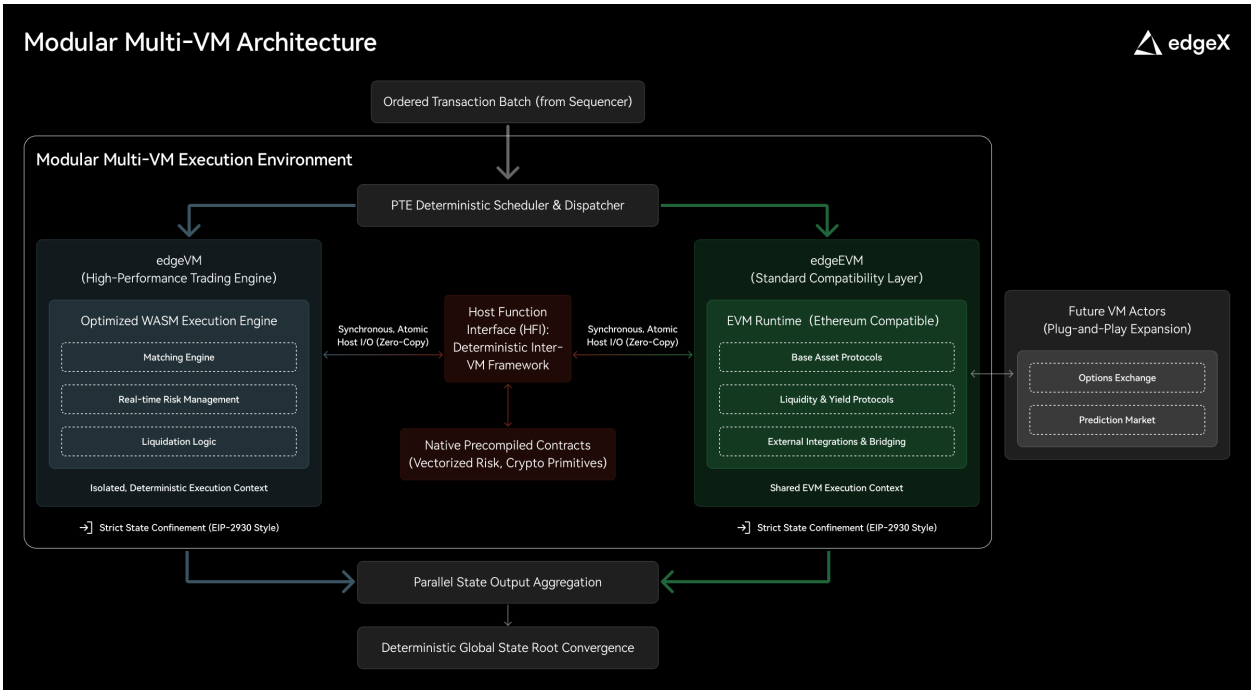

Figure 2: **Modular Multi-VM Execution Environment.**

The fundamental challenge in modular execution architecture is maintaining synchronous composability across heterogeneous environments. Standard approaches often fragment state into separate silos, relying on asynchronous bridging or message passing that introduces unacceptable latency and breaks atomic financial transactions.

EDGE Stack addresses this not with a bridge, but by establishing the Host Function Interface (HFI) as an intrinsic, rigorously defined interaction framework operating within a unified execution context.

Drawing inspiration from advanced integration paradigms like Arbitrum Stylus, edgeVM and edgeEVM do not operate as separate processes. Instead, they share a single block transition cycle. The HFI defines strict, deterministic ABI (Application Binary Interface) boundaries, allowing high-performance trading cores to perform direct, synchronous system calls to trigger standard EVM state transitions, and vice-versa.

3.3.1 I. Atomic Cross-Domain State Transitions

Consider a user depositing USDC collateral via a standard contract on edgeEVM. In EDGE Stack, this is not modeled as an asynchronous "transfer" event between chains. Instead, the deposit triggers a precisely defined Host I/O call that atomically mutates the user's margin balance state within the edgeVM trading engine in the exact same execution operation.

3.3.2 II. Zero-Copy Shared Memory Architecture

To mitigate the computational overhead typically associated with context switching between VMs, this framework utilizes Zero-Copy techniques wherever feasible. By passing data via validated pointers within a restricted shared memory heap—rather than incurring the cost of serialization and deserialization across boundaries—EDGE Stack achieves seamless, atomic composability while retaining strict determinism for complex financial operations.

3.4 Native Optimizations via Precompiled Contracts

While the edgeVM's architecture delivers exceptional performance for general matching logic, on-chain derivatives trading introduces a unique class of computational burdens that exceed the practical limits of any virtualized environment. The arithmetic density required for real-time margin calculations and the cryptographic load of high-frequency order submission create bottlenecks that threaten deterministic latency.

To overcome these domain-specific challenges, EDGE Stack incorporates specialized Precompiled Contracts. These are not generic accelerators, but native execution modules purposefully engineered to handle the heaviest mathematical primitives required by a perpetual futures exchange, embedded directly into the node's host infrastructure for zero-overhead invocation.

3.4.1 I. Solving the Derivatives Compute Burden: Vectorized Risk Engines

Unlike spot AMMs, perpetual exchanges demand continuous, complex mathematical re-evaluation. Maintaining systemic solvency requires real-time monitoring of cross-margined portfolios, mark price updates, and funding rate calculations across tens of thousands of active positions simultaneously.

In a parallelized system like EDGE Stack's PTE, a slow risk check becomes a "straggler task" that delays an entire execution wave. To ensure risk calculations never block order matching, EDGE Stack implements dedicated risk precompiles. This allows the system to recalculate the health of entire market segments in micro-seconds during extreme volatility, ensuring robust liquidation mechanisms without compromising exchange responsiveness.

3.4.2 II. Enabling HFT Ingress: Native Cryptographic Primitives

The primary users of EDGE Stack are institutional market makers who generate massive bursts of orders, cancellations, and modifications, particularly via FlashLane. Every single message requires cryptographic signature verification to prove authenticity.

At HFT scale, the cumulative CPU cycles required for verifying thousands of secp256k1 or ed25519 signatures per second become the dominant ingress bottleneck. EDGE Stack treats signature verification not as VM bytecode, but as a native protocol primitive. By offloading this intense workload to highly optimized native precompiles, the system ensures that the Sequencer can handle the bursty, high-volume traffic characteristic of mature financial markets without inducing latency queues at the gateway.

3.5 Plug-and-Play Expansion & Isolation

Building upon the extensibility framework outlined in Section 3.2, EDGE Stack realizes future-proof scalability through a concrete "VM Actor" implementation model. In this paradigm, distinct functional modules—whether a new

prediction market core or an experimental options engine—are treated not as tightly coupled code, but as independent, hot-swappable executive units.

3.5.1 I. The Plug-and-Play Deployment Lifecycle

A "VM Actor" technically manifests as a specialized, pre-compiled WASM artifact (or a set of standard EVM bytecodes) designed to adhere to a specific PTE interface. Adding new functionality follows a permissionless deployment lifecycle that avoids network forks:

1. **Deployment:** The new Actor's bytecode is deployed on-chain via a standard transaction to the edgeEVM compatibility layer.
2. **Registration:** The deployed contract registers itself with a central on-chain Protocol Registry, defining its operational parameters and requesting specific state access scopes.
3. **Activation:** Once approved (via governance processes or predefined criteria), the PTE Scheduler dynamically recognizes the new Actor ID and begins routing relevant transaction types to its dedicated execution context in the subsequent epoch.

3.5.2 II. Runtime Isolation Enforcement (The Sandbox)

While the PTE engine handles scheduling based on transaction dependencies, the ultimate security guarantee against cross-module state corruption lies in runtime enforcement. Isolation is cryptographically enforced by the Host Execution Environment.

Each registered VM Actor is assigned a rigid State Slot Manifest—a predefined scope of read/write permissions modeled on strict EIP-2930 principles [5]. When an actor attempts to execute an opcode that read or write state:

- **Interception:** The operation is intercepted by the underlying host environment before touching the global state.
- **Validation:** The target state slot is rigorously checked against the Actor's allowed Manifest.
- **Enforcement:** If the access attempt is outside the Actor's assigned sandbox (e.g., an experimental module attempting to alter the core BTC-PERP margin balance), the Host immediately reverts the operation.

This architectural firewall ensures that bugs in new, experimental modules remain hermetically sealed within their own boundaries, preserving the integrity of the core financial infrastructure.

4 Pillar II: Deterministic Parallel Transaction Execution (PTE)

4.1 The App-Specific Advantage: Deterministic Execution over Optimistic Models

General-purpose blockchains must optimize for maximum versatility across highly heterogeneous workloads, often leading to necessary architectural compromises. In the realm of parallel execution, leading general-purpose chains typically employ "Optimistic" models (e.g., Block-STM [4]). These engines assume state conflicts are rare, executing transactions in parallel initially and resolving any subsequent clashes via expensive rollbacks and re-executions. While effective for generic traffic with low contention, this approach hits a distinct performance ceiling during periods of high market volatility, introducing unpredictable latency spikes precisely when a financial exchange demands maximal stability.

As a specialized App-Specific Execution Layer, edge Stack possesses the strategic flexibility to move beyond generic optimizations. Unencumbered by the need to support arbitrary smart contracts, the architecture leverages the unique domain characteristics of high-frequency derivatives trading to implement a higher-ceiling "Deterministic" parallelization model.

Instead of reacting to conflicts dynamically at runtime, the PTE engine utilizes the known structure of derivative transactions—defined statically via Access Lists—to analyze dependencies before execution begins. This domain-aware approach ensures that transactions scheduled to run in parallel are guaranteed to succeed without conflict, unlocking zero re-execution overhead and the predictable, ultra-low latency profile that is unattainable in generalized environments.

4.2 Market-Sharded Execution via VM Actors

The architecture of EDGE Stack is founded on a critical domain insight: high-frequency perpetual futures trading exhibits a high degree of natural state isolation.

In generic DeFi protocols, a single transaction often interacts with multiple interconnected contracts (e.g., an AMM router routing a swap through three different liquidity pools). This creates complex, interleaved state dependencies that force sequential execution.

Conversely, in derivatives trading, a matching event occurring in the BTC-USDT central limit order book (CLOB) is functionally independent of the ETH-USDT order book. They share zero common state during the matching phase—they possess distinct order lists, distinct index price feeds, and distinct matching engine logic threads.

PTE leverages this inherent domain structure through Market-Sharded Execution, implemented using a rigorous Actor Model design pattern.

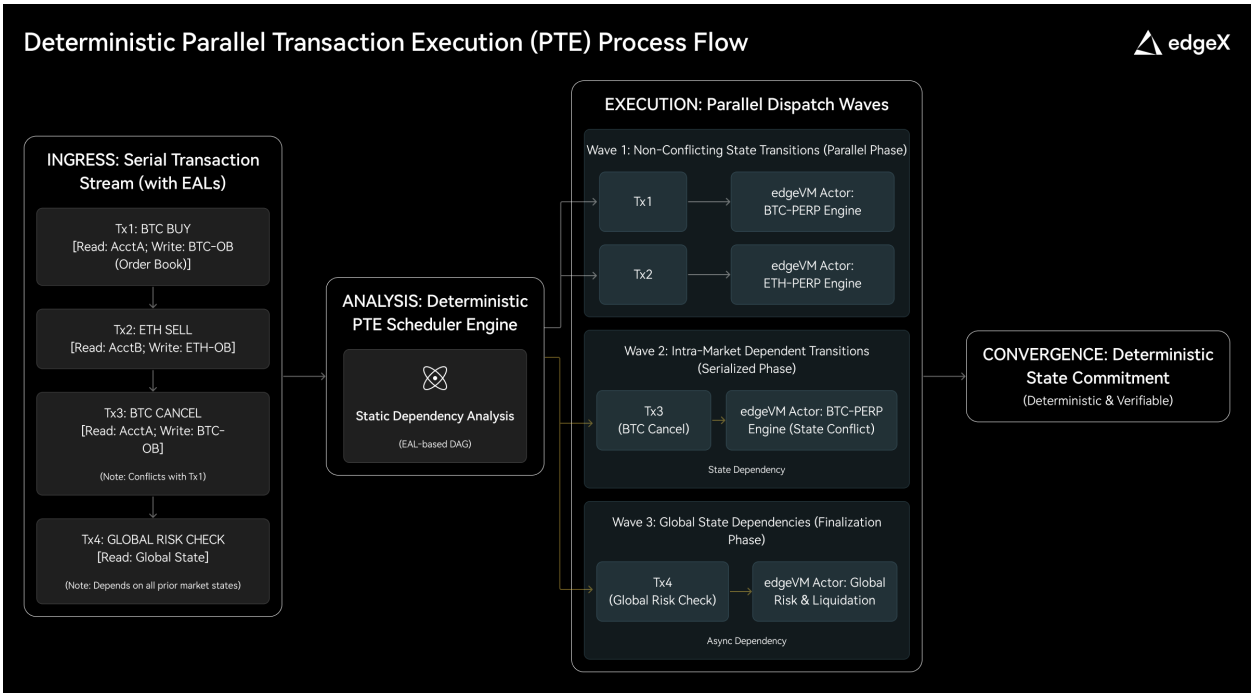

Figure 3: **Deterministic Parallel Transaction Execution (PTE) Flow.**

4.2.1 I. Autonomous VM Actors (The "Shared-Nothing" Approach)

In this framework, every listed derivative market (e.g., BTC-PERP, SOL-PERP) is instantiated as an autonomous VM Actor.

- **State Encapsulation:** Each Actor encapsulates its own mutable state (the CLOB data structure, local market parameters, recent trade history) and possesses a dedicated, single-threaded execution context.
- **Hard Boundaries:** Crucially, Actors operate in a "shared-nothing" architecture. They do not share memory and cannot directly read or modify each other's state. Communication between markets (if necessary, e.g., for cross-margin checks later in the lifecycle) occurs exclusively through deterministic asynchronous messaging handled by the PTE Scheduler, ensuring strict isolation boundaries.

4.2.2 II. Achieving Linear Scalability

The fundamental advantage of this shared-nothing Actor model is the realization of near-linear scalability for matching engines.

In monolithic blockchain systems, global state contention means that adding more CPU cores yields rapidly diminishing returns. The overhead of managing locks and thread synchronization across a unified state database eventually outweighs the benefits of extra compute power (a phenomenon dictated by Amdahl's Law [3]).

By contrast, because inter-market conflicts are virtually non-existent in EDGE Stack's matching phase, the system eliminates lock contention for order processing. Therefore, adding physical hardware threads translates directly to proportionally higher total system throughput (TPS). Ideally, a Validator node running 64 distinct Market Actors on a 64-core server approaches 64 times the aggregate matching throughput of a single-core setup.

In practical operation, while scaling remains highly efficient, CPU cores are not exclusively locked to individual VM Actors; they are dynamically shared with essential auxiliary system tasks, most notably the parallel Merklization processes detailed in Section 6.1. Even with this necessary resource sharing, this architecture allows EDGE Stack to absorb massive, simultaneous volatility spikes across diverse asset classes with minimal performance degradation compared to monolithic designs.

4.3 Conflict Resolution & Determinism Engine

Achieving high-throughput parallelism is relatively straightforward if one accepts eventual consistency or potential rollbacks. The supreme challenge in financial blockchain infrastructure, however, is achieving maximal parallelism while strictly maintaining serial semantics—ensuring the final state is mathematically identical to executing transactions one by one in their canonical order [15].

The core of the EDGE Stack PTE engine is its ability to rigorously resolve state contention without sacrificing speed. It transforms a serial stream of sequencer transactions into a highly optimized Directed Acyclic Graph (DAG) [7] of parallel tasks. This transformation relies on moving conflict detection from runtime to pre-execution analysis.

4.3.1 I. The Foundation: Extended Access Lists (EAL) and Formalized Conflict Logic

In standard EVM environments, a transaction's state impact is discovered dynamically during execution. A smart contract might call another contract based on variable on-chain conditions, making it impossible to predict precisely which storage slots will be read or written until the code actually runs. This dynamic nature is the primary antagonist of deterministic parallelism.

EDGE Stack introduces a paradigm shift requiring transactions to pre-declare their "state footprint" via Extended Access Lists (EALs). An EAL is a mandatory metadata layer specifying precisely the scope of state the transaction intends to access. By enforcing this static declaration, the PTE engine gains a "god's-eye view" of potential conflicts before execution begins.

Formalizing Parallel Compatibility Based on EALs, we can rigorously define the criteria for safe parallel execution. Let any transaction T_i be statically defined as a tuple of its Read Set (R_i) and Write Set (W_i):

$$T_i = (R_i, W_i) \tag{1}$$

For any two transactions T_i and T_j in an ordered batch (where $i < j$), they are parallel compatible ($T_i \parallel T_j$) if and only if their execution order does not affect the final state. This requires the intersection of their relevant state sets to be empty:

$$(W_i \cap R_j = \emptyset) \wedge (R_i \cap W_j = \emptyset) \wedge (W_i \cap W_j = \emptyset) \tag{2}$$

This ensures the absence of Read-after-Write (RAW), Write-after-Read (WAR), and Write-after-Write (WAW) conflicts within the same execution wave.

4.3.2 II. The Deterministic Scheduler Execution Waves

The Deterministic Scheduler is the brain of the PTE engine. It utilizes a greedy algorithm to reorganize the raw, ordered batch into a series of progressive "Execution Waves" based on the formal conflict logic defined above.

1. **Wave Construction:** The scheduler analyzes the batch sequentially, grouping the maximal set of non-conflicting transactions at the head of the queue into the current wave.
2. **Parallel Execution:** All transactions within a finalized wave are dispatched simultaneously across available VM Actors, executing in complete isolation.
3. **Dependency Resolution:** Transactions dependent on the state output of a prior wave must wait for that wave to complete atomically before being scheduled in a subsequent wave.

This wave-based approach maximizes hardware utilization by ensuring that at any given moment, every available CPU core is processing a valid, non-conflicting transaction.

4.3.3 III. The Deterministic Process Flow

The entire lifecycle of a transaction batch through the PTE engine follows a rigorous, reproducible path, distinct from optimistic execution models that rely on rollbacks:

1. **Ingestion & Validation:** The node receives an ordered batch and validates the format and EALs of all transactions.
2. **DAG Generation (Static Analysis):** Based purely on the input order and EALs, an in-memory dependency graph is constructed, mapping which transactions must precede others.
3. **Wave Scheduling:** The DAG is flattened into a deterministic sequence of parallel execution waves using the algorithm described above.
4. **Concurrent Execution & Convergence:** Waves are executed sequentially, with transactions inside each wave executing concurrently across VM Actors. The results merge to form the new finalized state root.

Because the entire scheduling logic is a pure function of the fixed input batch and the static EALs, the system guarantees strict adherence to serial semantics regardless of thread timing or hardware variations. This static approach provides the predictable performance profile demanded by high-frequency trading workloads.

4.4 Horizontal Scaling of Critical Subsystems

While Market-Sharded Execution effectively parallelizes order matching based on asset isolation, a robust perpetual futures exchange relies on computationally intensive auxiliary subsystems that often transcend individual market boundaries.

In monolithic architectures, functions like global portfolio risk calculation involve iterating through massive state arrays. During periods of high market volatility, the computational overhead of these calculations spikes precisely when trading volume is highest. If processed synchronously within the main execution loop, these subsystems become the dominant source of latency, "starving" the matching engine of resources.

EDGE Stack addresses this by decoupling these critical functions into autonomous, asynchronous VM Actors. This design allows subsystems that are computationally heavy but not strictly dependent on immediate order ordering to run in parallel to the core matching flow.

4.4.1 I. Asynchronous Risk Liquidation Engines

The safety and solvency of a derivatives exchange depend on two critical engines, both implemented as separate parallel Actors in EDGE Stack:

- **The Global Risk Engine Actor:** Maintaining system solvency requires real-time monitoring of every user's margin ratio across their entire portfolio. As prices fluctuate, the system must re-calculate the net asset value and maintenance margin requirements for tens of thousands of active accounts simultaneously. By isolating the Risk Engine into its own VM Actor, these massive computational tasks run asynchronously. The Risk Engine consumes a stream of completed trades and oracle price updates to update account health in the background, ensuring that heavy math never blocks a new order from entering the matching engine.
- **The Liquidation Execution Actor:** Liquidation is not a single event but a complex, multi-step state transition sequence: canceling open orders, seizing collateral, executing market orders to close positions, and updating insurance funds. If executed synchronously, a large liquidation cascade could stall a market for dozens of milliseconds. In EDGE Stack, once an account is flagged by the Risk Engine, the liquidation workflow is handed off to the dedicated Liquidation Actor. This ensures these complex operations are executed atomically and efficiently without degrading the latency for healthy market participants.

4.4.2 II. Dynamic Elasticity for Institutional Demands

The decoupling of logical Actors from physical hardware resources provides EDGE Stack with unprecedented elastic scalability. Because subsystems like the Risk Engine are isolated Actors communicating via deterministic messaging, they are not bound to the same physical CPU core as the matching engines.

- **Under Normal Load:** A single Validator node might run all Market Actors and Subsystem Actors on different cores of the same high-performance server.

- **Under Extreme Load:** As network activity spikes, the system architecture supports dynamic load balancing. Critical, compute-hungry Actors (like the Global Risk Engine executing during a crash) can be logically relocated to run on dedicated hardware threads or even separate physical server instances within the Validator's cluster.

This horizontal scaling capability ensures that EDGE Stack can absorb the massive throughput demands of institutional algorithmic traders without succumbing to resource contention that plagues monolithic designs.

5 Pillar III: FlashLane: Protocol-Native Transaction Prioritization

In standard monolithic blockchains, all transactions are treated equally. A high-frequency trader urgently trying to cancel a stale quote competes for the same block space and execution resources as a user claiming an airdrop or participating in governance voting. This lack of differentiation leads to Head-of-Line Blocking, where critical, latency-sensitive operations are delayed by bulky, non-urgent transactions, making predictable trading strategies impossible.

EDGE Stack addresses this fundamental flaw by introducing FlashLane, a protocol-native transaction prioritization mechanism. FlashLane is built on the premise that in financial markets, speed is not a luxury—it is a fundamental requirement for effective risk management.

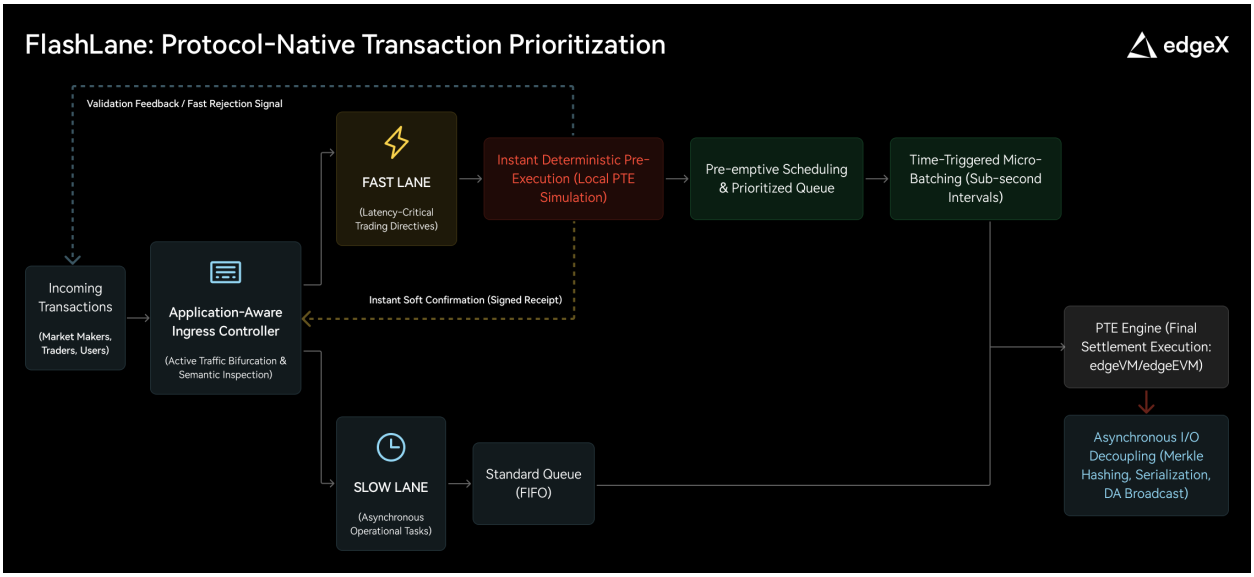

Figure 4: **FlashLane Protocol-Native Prioritization.**

5.1 Traffic Tiering: The Dual-Lane Architecture

In conventional blockchain architectures, the sequencer operates as a passive, monolithic FIFO (First-In, First-Out) queue. This undifferentiated approach is fundamentally flawed for high-performance exchanges, as a critical market-making quote is forced to wait behind unrelated, compute-heavy transactions (e.g., a complex governance vote initialization), resulting in unacceptable latency spikes known as Head-of-Line Blocking [22].

EDGE Stack fundamentally redesigns the sequencer ingress surface into an Application-Aware Ingress Controller. By performing real-time semantic inspection of incoming transaction types, the system actively bifurcates traffic into distinct processing lanes based on their operational urgency and resource profile.

5.1.1 I. The Fast Lane (Latency-Critical Path)

This lane acts as a dedicated, optimized express route reserved exclusively for atomic, high-frequency trading operations where deterministic latency is paramount for maintaining tight spreads and managing risk.

- **Scope of Operations:** Strictly limited to latency-sensitive trading directives, including:
  - New Order Placement (Limit, Market, IOC, FOK)

  - Order Cancellation
  - Order Modification/Replacement
- **Pre-emptive Scheduling Priority:** Transactions identified for the Fast Lane are not just prioritized; they are afforded pre-emptive scheduling status. The Sequencer ensures these transactions bypass pending Slow Lane operations in the ordering queue. Within the PTE engine, Fast Lane batches are allocated the immediate next available compute slots within their respective edgeVM market actors, ensuring minimal time-in-queue.

5.1.2 II. The Slow Lane (Asynchronous Settlement Path)

This lane handles the necessary operational backbone of the exchange that does not require microsecond-level execution guarantees. These transactions are typically characterized as being either state-heavy (complex EVM logic) or data-heavy (large calldata payloads for DA).

- **Scope of Operations:** Includes broadband operations such as:
  - Asset deposits and withdrawals (bridging operations).
  - Collateral management and margin adjustments.
  - Governance actions and staking operations.
  - General interactions with smart contracts deployed on edgeEVM.
- **Architectural Resource Isolation:** The critical innovation here is not merely the logical separation of traffic, but physical resource isolation. The processing of bulky Slow Lane transactions is handled strictly asynchronously to the main high-frequency matching loop.

Heavy state computations within the edgeEVM, alongside the intensive serialization of large data blobs required for ultimate finality and DA commitments, are offloaded to dedicated, separate thread pools. This architectural firewall ensures that a complex, resource-intensive operation in the Slow Lane can never "stop the world" or degrade the microsecond-level responsiveness of the Fast Lane matching engine.

5.2 FlashLane Commitment: Deterministic Soft Confirmations

The latency chasm between immutable on-chain settlement (Ethereum L1 finality, or even standard L2 consensus) and the requirements of high-frequency trading constitutes a critical market inefficiency. For institutional market makers, multi-second uncertainty regarding hedge execution is economically untenable, preventing accurate real-time inventory management and pricing model updates.

To bridge the gap between low latency and decentralized settlement assurances, EDGE Stack introduces FlashLane Commitment—a protocol mechanism designed to provide near-instant, cryptographically secured Soft Confirmations.

5.2.1 I. Instant Deterministic Pre-Execution

Upon ingress into the Fast Lane, the EDGE Stack Sequencer does not merely queue the transaction for future block inclusion. Instead, it immediately executes it against the current head state using a local instance of the Deterministic PTE Engine (as detailed in Section 4).

Because the PTE's state transition function is purely a product of the fixed input order and static Extended Access Lists (EALs)—devoid of runtime non-determinism like network jitter or probabilistic conflicts—this local execution accurately predicts the exact outcome of the transaction as it will appear in the next finalized batch. This is not an approximation, but a mathematically guaranteed pre-calculation.

5.2.2 II. The Cryptographic Commitment Receipt

Within sub-millisecond timeframes of ingress, the Sequencer issues a signed FlashLane Commitment back to the originating trader via WebSocket. This is not a vague acknowledgement, but a structured cryptographic receipt signed by the Sequencer's authoritative private key. The receipt contains precise execution metadata:

- The assigned Batch ID.
- The transaction's exact Ordered Index within that batch.
- The resulting Post-Execution State Delta (e.g., confirmed order fill quantity and price, updated account margin balance).

5.2.3 III. Economic Finality and Strategic Chaining

While ultimate protocol finality rests with Ethereum L1 settlement, the FlashLane Commitment provides traders with a mathematically rigorous assurance of inclusion ordering and execution outcome.

This mechanism offers what is known as "Economic Finality." Given the deterministic nature of the engine and the cryptographic signature of the sequencer, the risk of reordering is minimized to theoretical worst-case sequencer failure scenarios (which are further mitigated by decentralization roadmap plans). This high-confidence soft confirmation allows market makers to instantly update internal risk models and "chain" subsequent dependent strategies (e.g., placing a hedging trade on another venue milliseconds later) without waiting for block confirmation.

5.3 Optimized I/O and Micro-Batching

While Traffic Tiering and Deterministic Soft Confirmations optimize the logical processing flow, the physical integrity of the system's ultra-low latency relies on radically architecting the Sequencer's data ingestion and batch formation pipeline. In monolithic systems, these processes are often optimized for throughput (maximizing bytes per batch), creating a latency floor incompatible with high-frequency trading.

EDGE Stack reverses these optimization parameters within its Sequencer, prioritizing immediate "time-to-execution" above all else through two key internal mechanisms.

5.3.1 I. Time-Triggered Micro-Batching

To minimize the latency between transaction submission and execution finalization, EDGE Stack abandons size-based batching thresholds in favor of an aggressive Time-Triggered Micro-Batching strategy for Fast Lane traffic.

Instead of allowing transactions to idle in a mempool while waiting for a batch to fill—a common source of variable latency in general-purpose systems—the Sequencer finalizes and dispatches batches based on ultra-tight time intervals (on the order of milliseconds). This ensures that incoming market orders are packetized and sent to the PTE engine with minimal queuing delay, creating a continuous, high-frequency stream of executable micro-batches rather than infrequent, large blocks.

5.3.2 II. Asynchronous "Hot-Path" Isolation

Maintaining the predictable performance of the in-memory matching engine is paramount. In many architectures, the heavy I/O overhead associated with finalizing a batch—such as serializing data structures, recalculating large Merkle state tree differentials, and handling network disk persistence—can introduce brief "stop-the-world" pauses that disrupt matching continuity.

EDGE Stack enforces strict architectural isolation to ensure the critical "Hot Path" of order ingestion and matching is never blocked by these "Cold Path" administrative tasks.

- **Heavy Lifting Separation:** The physical processes of batch serialization, cryptographic hashing, and persistent storage logging are offloaded to dedicated, asynchronous thread pools.
- **Non-Blocking Execution:** This design ensures that the burdensome work of finalizing previous batches occurs completely in the background, without degrading the microsecond-level responsiveness of the Sequencer's core thread processing current incoming flow.

6 State Management & Security

In high-performance blockchain architectures, realizing ultra-high transaction throughput at the execution layer is merely the initial challenge. The predominant constraint affecting real-world scalability is the state management bottleneck—specifically, the immense cryptographic I/O overhead required to update the global Merkle Patricia Trie (MPT) [6] sequentially. A PTE engine processing thousands of trades per second is rendered ineffective if its underlying state commitments cannot keep pace.

EDGE Stack addresses this fundamental challenge through a holistic redesign of state lifecycle management. By implementing concurrent Merklization paired with deterministic state convergence [12, 13], the system ensures that cryptographic integrity aligns precisely with matching engine speed. Furthermore, by framing this high-performance state layer within a modular security architecture [14], EDGE Stack achieves centralized-level responsiveness while fully inheriting the trust-minimized verifiability essential to decentralized networks.

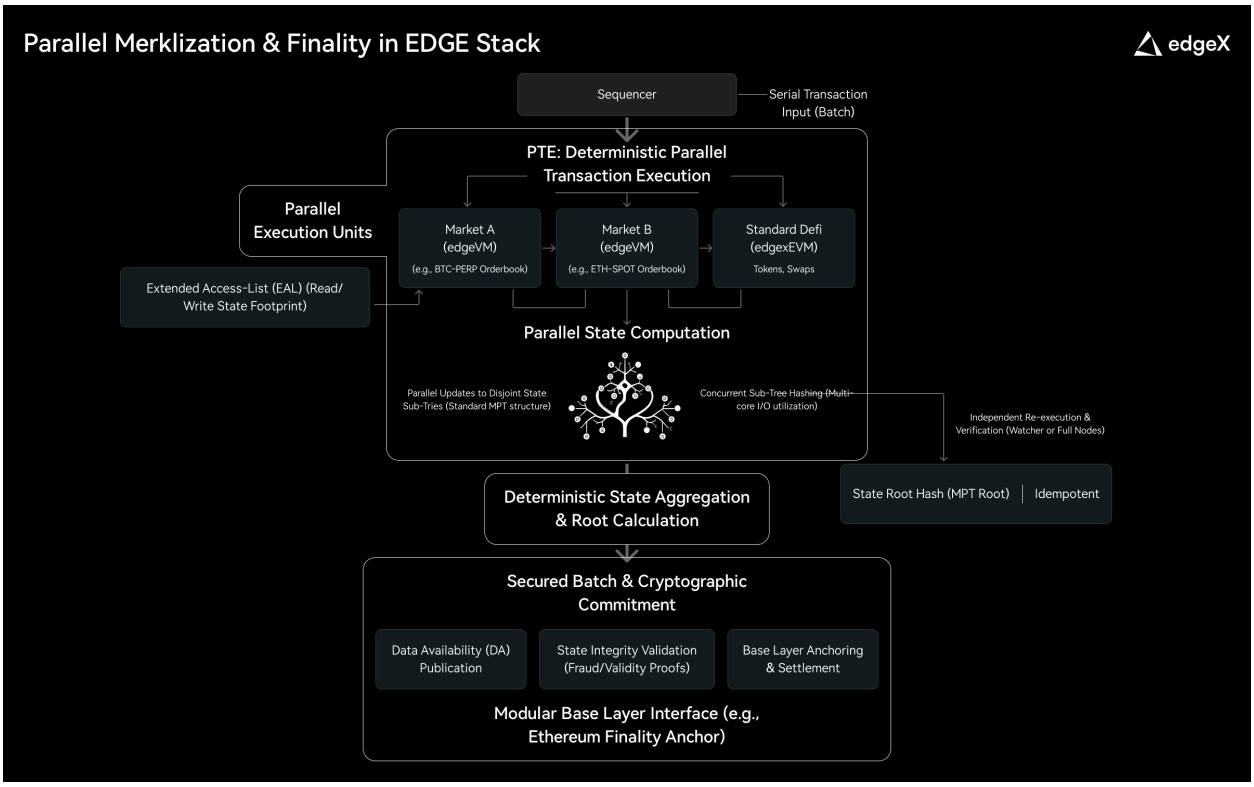

Figure 5: **State Lifecycle and Finality Architecture.**

6.1 Parallel Merklization: Breaking the Cryptographic I/O Bottleneck

6.1.1 I. The Monolithic Bottleneck: Serialized State Updates

In monolithic blockchain systems, state integrity is maintained via a singular, globally mutable data structure—typically using MPT. While the MPT is highly efficient for cryptographic verification, updating it is inherently serial. For every discrete state transition (e.g., modifying a balance or updating an order status), the system must acquire a write lock and recompute cryptographic hashes along the entire "path-to-root."

As transaction velocity accelerates into thousands per second, this repetitive rehashing induces severe write amplification. The data commitment process becomes a non-parallelizable serializing bottleneck governed by Amdahl's Law, throttling total system performance regardless of raw execution engine speed.

6.1.2 II. The EDGE Stack Solution: Parallelizing standard World State

EDGE Stack addresses this bottleneck without abandoning established standards. We maintain full architectural alignment with the Ethereum World State model, utilizing standard MPTs for both the global account state and individual contract storage states. This ensures maximum compatibility for future ecosystem migrations and tooling.

Instead of redesigning the data structure, EDGE Stack revolutionizes the process of updating it through Parallel Merklization, engineered to align precisely with the Deterministic PTE model.

This is achieved through three coordinated mechanisms:

- **Access-List Driven Parallel Hashing:** Leveraging the static analysis performed by the PTE engine (via Extended Access Lists), the system identifies beforehand which state slots will be modified in a given execution wave. Because PTE guarantees that concurrent transactions touch mathematically disjoint paths within the global trie, different CPU cores can be assigned the heavy lifting of recomputing the intermediate hashes along these separate paths simultaneously, without fear of race conditions.
- **Independent Contract Storage Tries:** Adhering to the standard Ethereum architecture, every contract account—including distinct VM Actors like the BTC-PERP engine—maintains its own independent Storage

Trie that holds its internal state. When different markets are active simultaneously, the system computes the updates for these independent Storage Tries in parallel threads. The new roots of these Storage Tries are then subsequently updated in the main Global Account Trie.

- **Asynchronous I/O Decoupling:** Crucially, this intensive cryptographic work is decoupled from the critical "hot path" of order matching and trade execution. The ingress of new orders and the core matching loop never block waiting for tree hashes to finalize or visualize to disk. State commitment and hash computation are treated as background processes, ensuring that cryptographic latency inherent to persistence does not degrade the real-time responsiveness of the exchange interface.

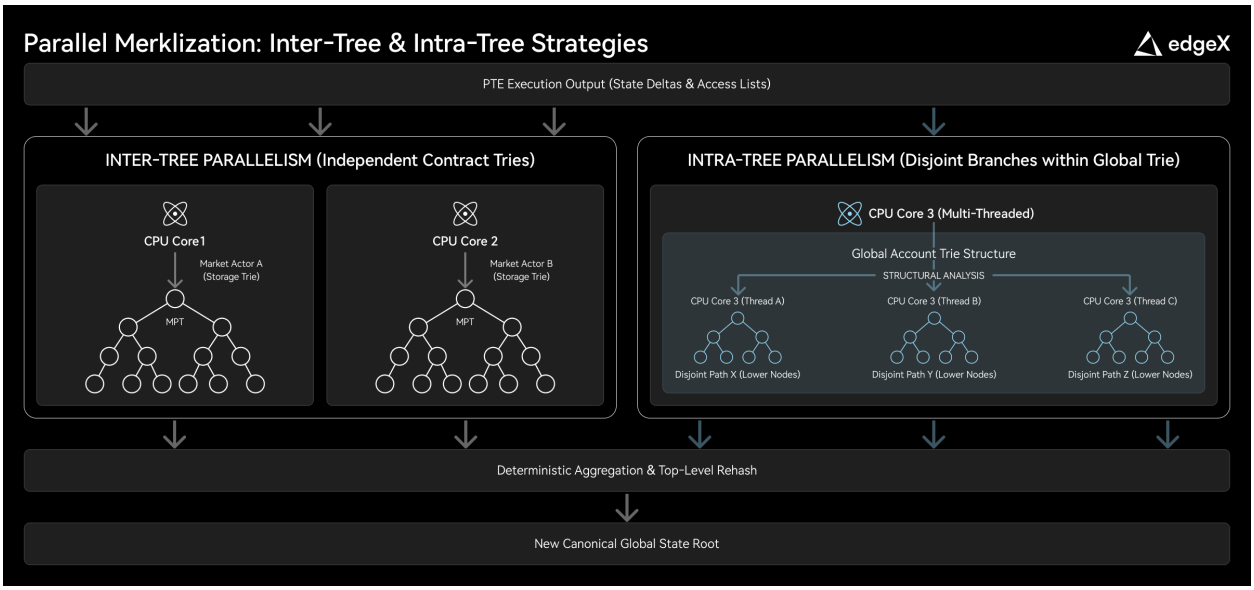

Figure 6: **Parallel Merklization Strategies.**

6.2 State Root Commitment: Deterministic Convergence

While execution and initial hashing occur via parallelized threads acting on disjoint state paths, adherence to the standard Ethereum World State paradigm requires a singular, canonical definition of truth at each block height. The fragmented outputs of parallel processing must be synthesized into one atomic commitment.

This section outlines the mechanism for achieving this convergence deterministically and efficiently.

6.2.1 I. The Foundation: Deterministic Causal Ordering

The prerequisite for safe state convergence is the rigorous ordering guarantee provided by the PTE engine (as detailed in Section 4). Although transactions execute physically in parallel, their logical **causal sequence** is fixed strictly by input order and dependency analysis before execution begins.

This means that the sequence in which resulting state changes are applied to the global trie is mathematically predetermined. There are no race conditions in the convergence phase because the ordering of updates is fixed, ensuring that reassembling the state is an entirely deterministic process readable by any observer.

6.2.2 II. The Aggregation Mechanism: Rapid Top-Level Hashing

Following the completion of an execution wave, the system enters the final aggregation phase. At this stage, the heavy lifting of deep hash recomputation has already been completed in parallel (via the mechanisms in 6.1).

- **Updating Contract Roots:** The new, finalized Storage Roots of independent VM Actors (e.g., the updated BTC-PERP market state trie) are handed off to the main state manager.
- **Bubbling Up:** These new intermediate roots are inserted as leaf updates into the main Global Account Trie.

- **Final Commitment:** Because only the paths leading to these specific contract accounts have changed, the system only needs to perform a rapid, serial rehash of the uppermost layers of the Global Account Trie.

This final operation produces a singular Global State Root. This hash forms the immutable cryptographic commitment representing the exact state of every user balance, open order, and risk parameter across the entire exchange ecosystem at that specific block height, ready for settlement.

6.3 Modular Security Architecture: The Anchor of Trust

While EDGE Stack operates as a high-performance, sovereign execution environment, its security model is founded on the Modular Blockchain Thesis. Rather than bootstrapping an isolated validator set for ultimate security in its initial iteration, EDGE Stack decouples execution speed from settlement security, outsourcing consensus and data availability to the most robust infrastructure layers available.

This architecture ensures that the exchange's security properties are anchored not to the honesty of the operator, but to the cryptoeconomic guarantees of a robust base layer (e.g., Ethereum).

6.3.1 I. The Pluggable Settlement Anchor

The deterministic Global State Roots produced by the EDGE Stack sequencer serve as cryptographic checkpoints. These roots are periodically published to a modular settlement layer. This anchoring ensures that the finalized state of the exchange inherits the immutable finality properties of the chosen base layer, providing an objective, tamper-proof history of all state transitions.

6.3.2 II. The Data Availability Foundation

Security relies fundamentally on data visibility. EDGE Stack broadcasts the compressed, raw transaction inputs necessary to reconstruct the state to a high-integrity DA layer. By ensuring this data is publicly accessible and resistant to censorship on the base layer, the system guarantees the raw materials for verification are always available, regardless of the sequencer's operational status.

6.4 Transparency & Verifiability: The "Don't Trust, Verify" Mechanism

The ultimate safeguard in any decentralized system is permissionless auditing. While Section 6.3 outlines where the data and commitments are stored, this section details how participants utilize that infrastructure to ensure integrity.

EDGE Stack enables a regime of radical transparency by combining the previously described Modular Data Availability with its rollback-free, deterministic PTE engine. This combination ensures that high performance never comes at the cost of verifiable correctness.

- **I. Permissionless Auditing via Watcher Nodes:** Any external participant—institutional market makers, auditors, or community researchers—can operate a Watcher Node. By independently downloading the raw transaction batches stored on the DA layer and feeding them into a local, verified instance of the EDGE Stack software, the node will reproduce the exact sequence of state transitions active on the main network.
- **II. Cryptographic Accountability:** Because the execution path is rigidly fixed via pre-execution analysis (avoiding nondeterministic runtime conflict resolution), a Watcher Node must arrive at a Global State Root identical to the one published by the Sequencer on the settlement layer.

If a locally computed root differs mathematically from the published root, it constitutes immutable cryptographic evidence of malfeasance by the Sequencer. This mathematical certainty forms the basis of trustless accountability protocols, such as fraud proofs, ensuring the system remains honest even without a trusted operator.

7 Conclusion

EDGE Stack introduces a paradigm shift in blockchain infrastructure designed to resolve structural contradictions hindering high-frequency decentralized finance. By reimagining the stack not as a general-purpose blockchain, but as an App-Specific Execution Layer engineered for trading, it bridges the immense performance gap between centralized venues and on-chain environments without compromising core Web3 principles.

Unlike monolithic designs that force financial logic to compete for resources on a single global state machine, EDGE Stack decouples execution through a novel Modular Multi-VM architecture. It replaces optimistic models with a

Deterministic PTE engine, leveraging market-sharded state to achieve linear scalability and zero re-execution overhead. Furthermore, by implementing FlashLane, a protocol-native transaction prioritization mechanism, the system delivers the sub-millisecond latency required by institutional market makers.

Ultimately, EDGE Stack establishes a non-custodial, fully verifiable, and ultra-low-latency Sovereign Execution Environment. This architecture redefines the boundaries of on-chain financial infrastructure, positioning EDGE Stack as the foundational layer for the next generation of global market structure.

8 Future Work

At its core, the EDGE Stack is designed as a resilient foundation for a diverse array of financial operations. While its initial deployment focuses on advancing perpetual futures trading, the underlying architecture—specifically the PTE engine and modular runtime environment—is purposefully engineered to generalize across multiple asset classes and market types.

The vision for EDGE Stack extends beyond its current configuration. We are actively exploring several strategic avenues to broaden its financial capabilities and integrate emerging technologies:

- **Modular & Sovereign Settlement Evolution:** We are researching pathways to evolve EDGE Stack's security architecture. This involves leveraging its modular design to decouple the settlement layer fully, allowing the system to anchor its state to alternative high-throughput Data Availability (DA) layers or even establish itself as a sovereign network with its own consensus in the future. This flexibility ensures long-term adaptability to the evolving blockchain landscape.
- **Institutional Privacy Zones:** Recognizing the needs of large-scale participants, we plan to introduce optional privacy-preserving trading zones. Leveraging technologies like Zero-Knowledge Proofs (ZKP) or Trusted Execution Environments (TEE) within dedicated VM Actors, these zones will offer features such as encrypted order books to mitigate information leakage and front-running, further enhancing market integrity for institutional capital.
- **Cross-Chain Interoperability Standards:** We are committed to integrating native, trustless interoperability standards, such as Circle's Cross-Chain Transfer Protocol (CCTP) [16], to enable seamless liquidity migration and unified asset management across the multi-chain ecosystem.
- **Native AI-Driven Financial Intelligence:** The high-performance, deterministic execution environment of EDGE Stack is uniquely suited for integrating machine learning directly into the transaction lifecycle. We are exploring dedicated "AI VM Actors" optimized for on-chain inference (e.g., running specialized ML models specifically for finance). This will enable a new class of autonomous financial agents capable of executing sophisticated, data-driven trading strategies and real-time dynamic risk assessment directly at the execution edge, co-located with market liquidity.

Through these advancements, EDGE Stack aims to evolve into a comprehensive, multi-paradigm financial infrastructure that combines highly efficient, purpose-built trading primitives with privacy, interoperability, and native intelligent capabilities.

Acknowledgments

The development of EDGE Stack has benefited significantly from ongoing dialogue with leading experts across the high-performance computing and decentralized finance communities. We are grateful for the critical feedback and technical insights regarding parallel execution models, state machine architecture, and market microstructure design that have shaped this paper and our underlying research.

We would particularly like to acknowledge the pioneering work done by the teams at Offchain Labs (Arbitrum) and the broader research community investigating next-generation execution environments. Their contributions to the field have been instrumental in defining the problem space and inspiring our approach to App-Specific Execution Layers.

References

[1] Offchain Labs. (2024). "Stylus: Rust, C, and C++ on Arbitrum." *Arbitrum Documentation*. `https://docs.arbitrum.io/stylus/gentle-introduction`

[2] Wood, G. (2016). "Polkadot: Vision for a heterogeneous multi-chain framework." *White Paper*.

[3] Amdahl, G. M. (1967). "Validity of the single processor approach to achieving large scale computing capabilities." *In Proceedings of the April 18-20, 1967, spring joint computer conference* (pp. 483-485).

[4] Gelashvili, R., Spiegelman, A., Xiang, Z., et al. (2023). "Block-stm: Scaling blockchain execution by turning ordering curse to a performance blessing." *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming* (pp. 232-244).

[5] Buterin, V., & Swende, M. (2020). "EIP-2930: Optional access lists." *Ethereum Improvement Proposals*. `https://eips.ethereum.org/EIPS/eip-2930`

[6] Ethereum Foundation. (n.d.). "Merkle Patricia Trie." *Ethereum Developers Documentation*. `https://ethereum.org/en/developers/docs/data-structures-and-encoding/patricia-merkle-trie/`

[7] Shrier, I., & Platt, R. W. (2008). "Reducing bias through directed acyclic graphs." *BMC Medical Research Methodology*, 8(1), 70.

[8] Alchemy. (2023). "Modular vs. Monolithic Blockchains." *Alchemy Overviews*. `https://www.alchemy.com/overviews/modular-vs-monolithic-blockchains`

[9] Buterin, V. (2021). "An Incomplete Guide to Rollups." *Vitalik.ca*. `https://vitalik.eth.limo/general/2021/01/05/rollup.html`

[10] Celestia. (n.d.). "Data Availability Layer." *Celestia Documentation*. `https://docs.celestia.org/learn/how-celestia-works/data-availability-layer`

[11] Dodmane R, KR R, NS K R, et al. Blockchain-based automated market makers for a decentralized stock exchange[J]. Information, 2023, 14(5): 280.

[12] Zhang, F., Cecchetti, E., Croman, K., Juels, A., & Shi, E. (2016). "Town Crier: An Authenticated Data Feed for Smart Contracts." *In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 270-282).

[13] Herlihy, M., & Moss, J. E. B. (1993). "Transactional memory: Architectural support for lock-free data structures." *ACM SIGARCH Computer Architecture News*, 21(2), 289-300.

[14] Al-Bassam, M., Sonnino, A., Bano, S., Hrycyszyn, D., & Danezis, G. (2019). "LazyLedger: A Distributed Data Availability Ledger with Client-Side Validation." *arXiv preprint arXiv:1905.09274*.

[15] Papadimitriou, C. H. (1979). "The serializability of concurrent database updates." *Journal of the ACM (JACM)*, 26(4), 631-653.

[16] Circle Internet Financial. (2023). "Cross-Chain Transfer Protocol (CCTP) Technical Documentation." *Circle Developer Docs*.

[17] Daian, P., Goldfeder, S., Kell, T., Li, Y., Zhao, X., Bentov, I., ... & Juels, A. (2020). "Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability." *In 2020 IEEE Symposium on Security and Privacy (SP)* (pp. 910-927). IEEE.

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

[19] Dijkstra, E. W. (1982). "On the role of scientific thought." *Selected writings on computing: a personal perspective* (pp. 60-66). Springer, New York, NY.

[20] Buterin, V. (2022). "The different types of ZK-EVMs." *Vitalik.ca*. `https://vitalik.eth.limo/general/2022/08/04/zkevm.html`

[21] Croman, K., Decker, C., Eyal, I., Gencer, A. E., Juels, A., Kosba, A., ... & Wattenhofer, R. (2016). "On scaling decentralized blockchains." *In International Conference on Financial Cryptography and Data Security* (pp. 106-125). Springer, Berlin, Heidelberg.

[22] Scharf, M., & Kiesel, S. (2006). "Head-of-line Blocking in TCP and SCTP: Analysis and Measurements." *In IEEE Globecom 2006* (pp. 1-5). IEEE.

[23] Mohan V. Automated market makers and decentralized exchanges: a DeFi primer[J]. Financial Innovation, 2022, 8(1): 20.