PROTOCOL LAYER // EXECUTION

CLC Labs

Eliminating structural waste in AI agent execution.

Defining execution-level continuation at the runtime layer.

Sean King — Founder & Chief Architect | clclabs.ai

THE THESIS

The Recomputation Crisis

Agentic workflows fail at scale because execution is stateless across steps.

Statelessness is Structural Waste

Zero Execution Memory

Current inference engines treat every step as a fresh request, discarding accepted execution state across multi-step workflows.

The Quadratic Tax

As workflows deepen, stable context is re-encoded on every step, creating a quadratic compute and latency tax that model cost curves cannot fix.

The Economic Constraint

40%

Inference Waste per Step

Observed in multi-step agent workflows with verifier loops and retries.

Buyers Feel the Pain

Infrastructure and platform teams hit the compute wall first. GPU scarcity and multi-second latency SLAs turn recomputation into a hard production failure.

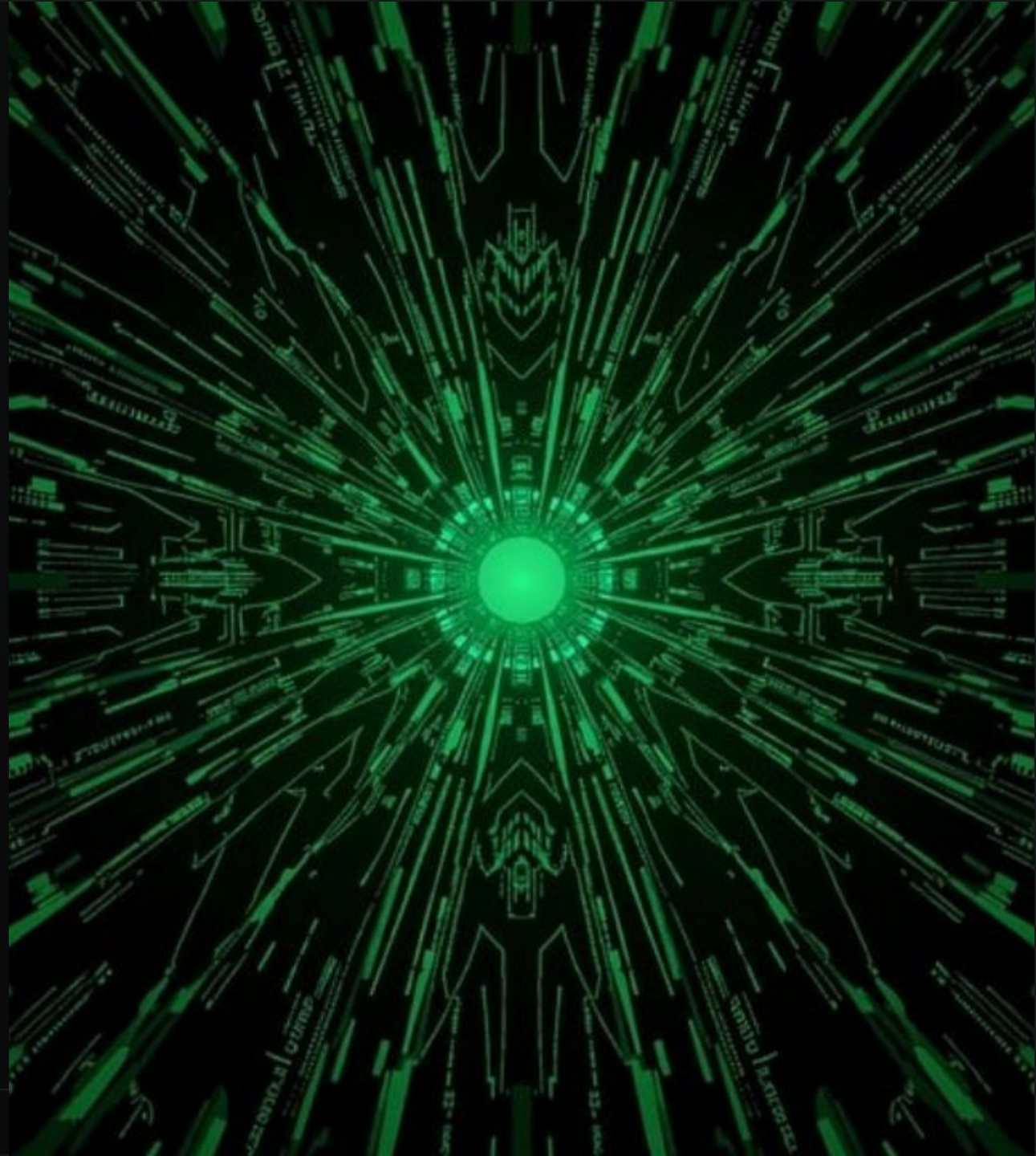
THE SOLUTION

Latent Execution (LE): Enforcing Continuation

Persistence by Design

Reuse verified AI work. Stop replaying history.

LE is a protocol primitive operating at the runtime execution boundary. It enforces Continuation vs. Replay, omitting redundant work while preserving output correctness.



The LE Runtime Engine

Agent Logic

↓

CLC Execution Boundary

↓

Inference Engine

Engine Agnostic

Drop-in execution layer compatible with vLLM, SGLang, and standard inference runtimes. Operates below agent logic and above the inference engine.

Secure State

Production-safe execution state with no raw KV export and strict hardware isolation by design.

Infra Standard

Infrastructure layer — not a tool. Defines how agent execution state is persisted and resumed across steps. Removing it reverts workflows to replay.

Proof: TFT + Energy Wins (IP-safe Benchmark)

Environment: A40 RunPod | Engine: SGLang | Model: OLMo-3-7B-Instruct

Prefill-dominant workload

Prefill-dominant workload; decode is trivial (4 tokens).

Median TTFT: 234.0 → 61.3 ms - 3.8x faster

Median E2E: 314.4 → 139.4 ms - 2.3x faster

Energy: 11.265 → 3.636 kJ - 3.1x lower

Churn-adversarial

120 steps, 120 unique prompt hashes

Median TTFT: 239.7 → 64.3 ms - 3.7x faster

Median E2E: 318.4 → 149.6 ms - 2.1x faster

Energy: 11.402 → 4.157 kJ - 2.7x lower

What this proves

- Demonstrates a **stable-prefix execution boundary** can preserve **engine-native cache reuse** across many continuations (**120 steps**).
- Shows the boundary remains effective under **adversarial churn** (**120 unique prompts**).
- Shows **TTFT and energy** improvements in **prefill-dominant** workloads where **prompt processing** is the bottleneck.

What this does not prove

- Does **not** claim universal end-to-end speedups on **decode-dominant** workloads (long generations).
- Does **not** claim portability to engines **without** session/prefix-cache primitives.
- Does **not** claim production durability or **security hardening** beyond this controlled benchmark.

Replay breaks at depth

Identical tasks · identical semantics · execution under churn

Depth Amplifies Replay

In multi-step agent workflows, stable context is re-encoded on every step. As depth increases, replay — not reasoning — dominates latency and cost.

Caching Is Insufficient

Prefix and KV caching optimize token reuse, but fail under branching, retries, and verifier loops because they cannot distinguish continuation from replay.

Execution Continuity Is Required

Preventing recomputation requires enforcing continuation at the runtime boundary — preserving accepted execution state across steps rather than reprocessing history.

Early adversarial benchmarks designed to expose execution failure modes, not peak throughput.

The irreducible architectural gap

Inference engines are designed to optimize throughput (tokens/sec), not execution semantics across steps. Governing continuation versus replay requires owning the runtime boundary between the model and the environment — a layer engines cannot absorb without breaking their core abstraction. CLC owns execution continuity, creating a structural lock on agent state that regresses to replay if removed.

The Founder

Sean King

CHIEF ARCHITECT

Former Infrastructure Lead at Fortune 5 enterprise.

Former Infrastructure Lead at a Fortune-5 enterprise. Led large-scale, high-availability systems under strict SLAs, owning vendor strategy and multi-year infrastructure spend. Bringing execution discipline from regulated infrastructure into agentic AI runtimes.

Roadmap to Standardization

PHASE 1

Lock IP & Adversarial Benchmarks

PHASE 2

Paid Partner Pilots (Infra Teams)

PHASE 3

Runtime Hardening & Multi-tenancy

PHASE 4

GA for Scale-ups & Agent Builders

Early phases prioritize correctness and proof over distribution and scale.

THE CAPITAL

The Raise

$250k SAFE — buying proof, not scale

Use of funds:

- Lock core execution IP
- Publish adversarial, third-party-verifiable benchmarks
- Validate with 3–5 infrastructure design partners

Contact

CLC Labs

Sean King — Founder & CEO

https://www.clclabs.ai

sean@clclabs.ai