Spin Made WebAssembly Serverless. The Functions Are Still Trigger-Based.

by Nick Clark | Published March 28, 2026 | PDF

Spin provides a developer-friendly framework for building serverless WebAssembly applications that respond to HTTP requests, Redis messages, and other triggers. The developer experience is excellent. But Spin functions are trigger-based: they execute when an event arrives and terminate when processing completes. They do not maintain persistent state between invocations, carry their own execution cycle, or self-evaluate. The gap is between serverless function hosting and memory-resident objects that execute from their own state.

Spin's developer experience and WebAssembly-native serverless model provide genuine improvements in function hosting. The gap described here is about the execution model.

Trigger-based execution without continuity

A Spin application defines handlers for triggers: HTTP routes, Redis channels, or timers. When a trigger fires, Spin instantiates the WebAssembly module, calls the handler, and discards the instance after completion. There is no state continuity between invocations. Each invocation starts fresh.

Memory-resident execution requires continuity: state that persists across cycles, memory that accumulates, and governance that validates continuously. Spin's model is optimized for stateless triggers, not stateful self-execution.

Key-value storage without governed memory

Spin provides key-value storage for persisting data between invocations. This addresses the statefulness gap at the infrastructure level. But key-value storage is a data store, not governed memory. There is no schema, no lineage, no governance on mutations.

What memory-resident execution provides

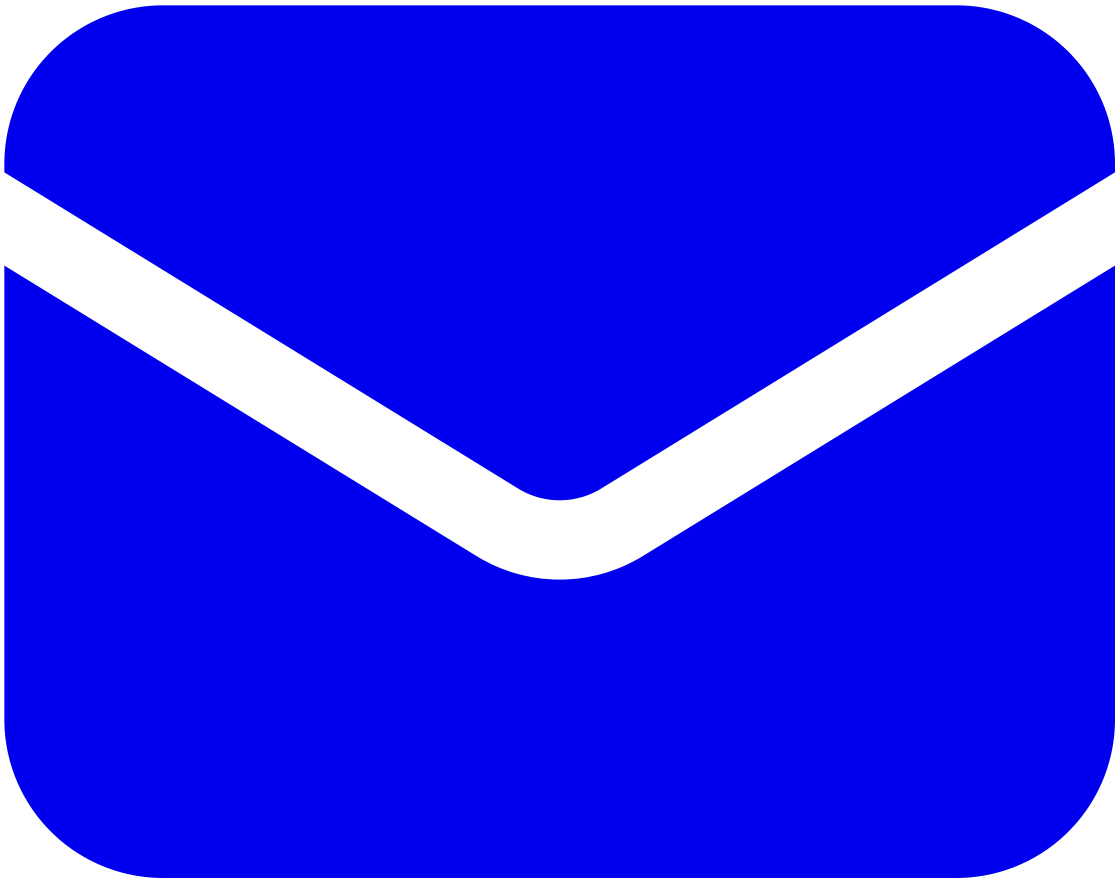
Memory-resident execution objects maintain persistent governed memory across execution cycles. They self-evaluate on each cycle, not just when triggered. Their memory includes lineage tracking and governance validation. Spin's efficient WebAssembly runtime could serve as the compute substrate. Memory-resident execution would add continuous self-evaluation and governed state above the trigger model.

Memory-Resident Execution All 21 steps →

Persistent objects that execute without orchestration.

Patent
US 19/538,221 · filed
Primary Technical Disclosure
○ Memory-Resident Execution: Persistent Semantic Objects Without Orchestration
Secondary Technical
○ Six-Action Execution Evaluation Cycle: Parse, Evaluate, Select at Every Node○ Cognition-Authority-Execution Separation: Reasoning Cannot Authorize Action○ Dormancy as First-Class Execution State: Valid Suspension Without Failure○ Semantic Backoff: Retry Pacing From Execution Outcomes Rather Than Fixed Timers○ Wake Triggers for Dormancy Exit: Explicit Reentry Conditions in Memory○ Persistent Polling Behavior: Autonomous Condition Evaluation Without Schedulers○ Intent Refinement During Execution: Adaptive Objectives Without Re-Instantiation○ Compositional Execution Through Recursive Delegation: Parent-Child Lineage Tracking○ Negative Capability Signals: Recording What Cannot Be Done as Structured Constraint○ Swarm-Based Execution Emergence: Coordinated Behavior Without Centralized Control○ Latency and Failure as Semantic Signals: Structured Inputs From Adverse Conditions○ LLM as Advisory Execution Node: Inference Without Authority Over Agent State○ Append-Only Memory Field: Complete Execution Lineage Through Immutable Records
Applications (General)
○ Serverless Execution Without Cold Starts or State Loss○ Long-Running Autonomous Workflows Without External Orchestration○ Drone Operations Surviving Disconnection○ Deep Space Agent Execution Without Ground Control○ Underwater Robotic Operations Without Connectivity○ Rural Healthcare Agents Surviving Intermittent Connectivity○ Operations in Infrastructure-Destroyed Environments○ Offline Financial Transaction Agents
Applications (Specific)
○ Cloudflare Durable Objects Made State Local. The Objects Still Need Orchestration.○ Azure Service Fabric Actors Are Addressable. They Are Not Autonomous.○ Akka Perfected the Actor Model. Actors Still React Instead of Self-Execute.○ Orleans Made Virtual Actors Practical. The Actors Still Execute on Request.○ Dapr Provides a Sidecar Runtime for Microservices. The Services Still Need External Orchestration.○ wasmCloud Runs WebAssembly Actors. The Actors Wait for Messages.● Spin Made WebAssembly Serverless. The Functions Are Still Trigger-Based.○ Fermyon Built the WebAssembly Cloud. The Cloud Hosts Functions, Not Self-Executing Objects.○ Fly Machines Made Micro-VMs Fast. The VMs Still Need External Orchestration.○ Railway Simplified Application Deployment. The Applications Still Depend on External Execution Triggers.
Memory-Resident Execution overview →
AQ
deterministic
autonomy

Last updated: 2026-03-03



-
- nick@qu3ry.net
- 72 28 14 36 01

Invented by Nick Clark | Founding Investors: Devin Wilkie