Dapr Provides a Sidecar Runtime for Microservices. The Services Still Need External Orchestration.

by Nick Clark | Published March 28, 2026 | PDF

Dapr provides a portable, event-driven runtime for building microservices with state management, pub/sub, and service invocation through a sidecar architecture. The building blocks simplify distributed application development. But Dapr services are request-driven: they respond to invocations and events. They do not self-execute from their own memory, self-evaluate their state, or autonomously decide to act. The gap is between a distributed runtime and memory-resident objects that carry their own execution cycle.

Dapr's sidecar architecture and building blocks genuinely simplify distributed application development. The gap described here is about the execution model, not about runtime capabilities.

Sidecar runtime without self-execution

Dapr attaches a sidecar process to each microservice, providing state management, pub/sub, and service invocation through a consistent API. The sidecar handles infrastructure concerns. But the microservice still executes in response to external events: HTTP requests, pub/sub messages, or bindings. It does not carry its own execution cycle.

A memory-resident execution object evaluates its own state on each cycle and decides whether to act. A Dapr service waits to be told to act.

State management without governed memory

Dapr's state management provides key-value storage with configurable consistency and backends. This is useful infrastructure. But the state is a data store, not governed memory. There is no schema for what the state means, no governance on state transitions, and no lineage tracking how state evolved.

What memory-resident execution provides

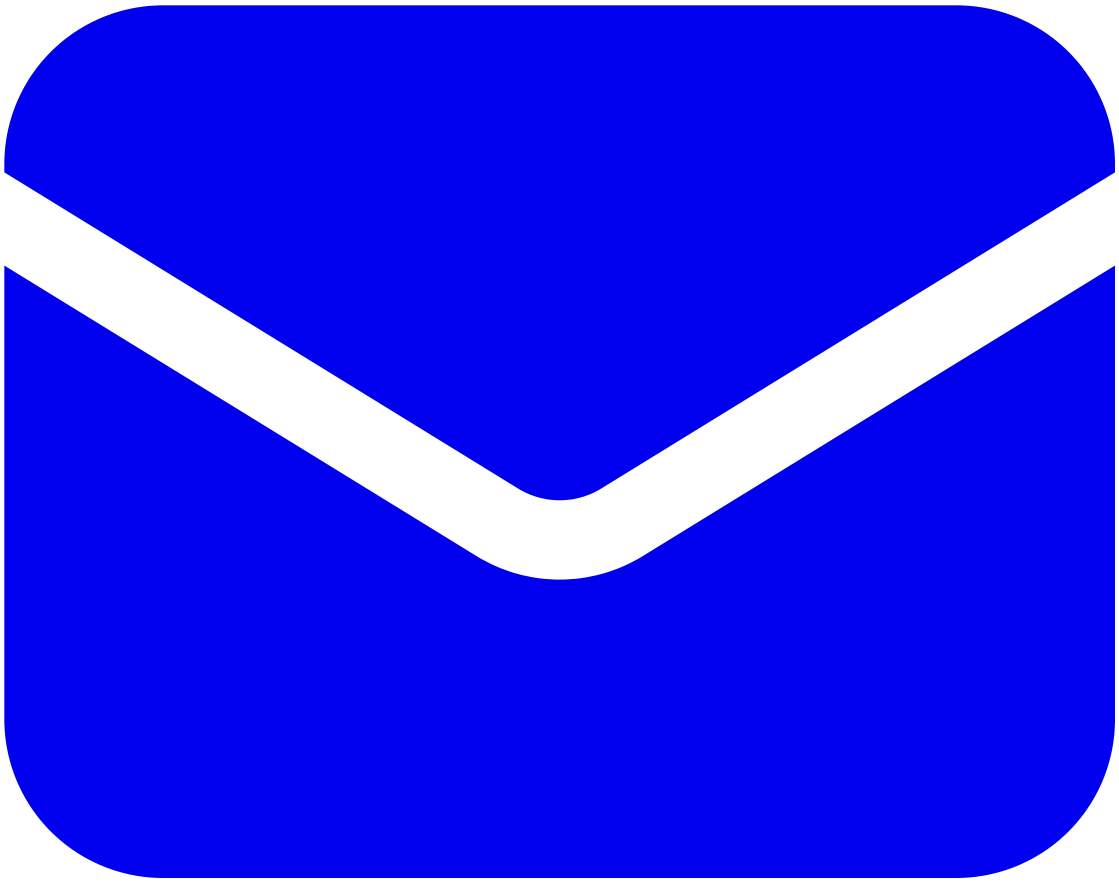
Memory-resident execution objects carry their own execution state, self-evaluate on each cycle, and maintain governed memory with lineage. The object does not wait for external events to decide to act. It continuously evaluates its state and acts when conditions are met. Dapr's state management and pub/sub could provide infrastructure for memory-resident objects. The self-execution cycle and governed memory would be intrinsic to the object.

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