Synapse Core API Refactoring

Migrating to an Event-Driven Microservices Architecture

Alex Mercer Priya Sharma

Synapse Architecture Committee ▪ Nexa Cloud Division

August 2026

Agenda

Meeting Roadmap

Legacy Architecture Bottlenecks

Proposed Microservices Architecture

Performance Benchmarks

Migration Strategy

The Legacy Synapse Core Monolith

Current Production Bottlenecks

The existing **Synapse Core Monolith** serves all transactional traffic. Under peak loads, it exhibits severe scaling constraints [1].

- **Tight Coupling:** Shared database schema blocks independent module deployments.
- **Resource Contention:** High-compute PDF generation starves low-latency payment endpoints.
- **Single Point of Failure:** Memory leaks in any module can crash the entire JVM application.

Production Incident Summary (Q2 2026)

During the Apex-Sale campaign, a spike in checkout notifications resulted in a 4-hour outage of the core API, costing approximately $1.2M in deferred revenue.

Target Event-Driven Architecture

Services & Decoupled Communication

We propose decomposing Synapse Core into decoupled services communicating asynchronously via Apache Kafka.

Architecture Components:

- **API Gateway:** Handles routing, auth, and rate limiting.
- **Order Service:** Checkout orchestration using Postgres.
- **Notify Service:** Async notification dispatch (email/SMS).
- **Kafka Cluster:** Log-centric event broker for services.

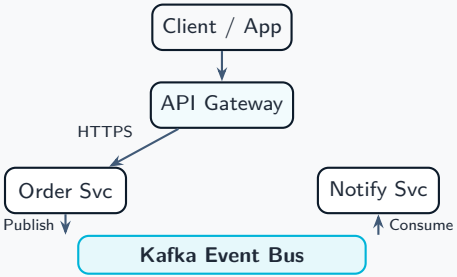

Figure 1: Event-driven asynchronous model.

Performance Benchmarks

Throughput and Resource Efficiency

Load testing was conducted on a simulated 8-node Kubernetes cluster at Nexa Labs [2].

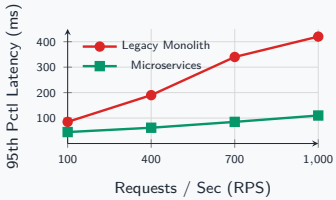

Figure 2: Latency under load scaling.

Key Measurements:

- **Peak Latency:** Monolith latency degrades exponentially beyond 600 RPS due to DB pool locking.
- **Throughput Limit:** Microservices sustained 1,000+ RPS with flat sub-120ms latency.

Metric	Monolith	Proposed	Gain
Max RPS	650	2,200	+238%
DB CPU %	88%	32%	-63.6%
Deploy Time	45m	4m	-91.1%

Migration Strategy & Rollout Plan

Risk Mitigation & Phasing

To prevent service disruption, we will execute a 3-stage rollout using the Strangler Fig pattern.

Phase 1: Read-Only Replication (Weeks 1–4)

Route search traffic to the new Notify service while writing to both database instances. Verify consistency.

Phase 2: Write Shadowing (Weeks 5–8)

Shadow writes to Order Service and validate transaction state differences asynchronously. No live customer dependencies.

Phase 3: Direct Cutover (Week 9)

Point DNS to the API Gateway. Keep legacy monolith standby instance operational for 7 days as rollback target.

References I

[1] Marcus Chen and Elena Rostova. “Deconstructing the Legacy Core: Pain points of monolith schemas”. In: *Synapse Technical Review* 8.3 (2024), pp. 45–59.

[2] Alex Mercer and Priya Sharma. “Scaling Transaction Processing with Event-Driven Architectures”. In: *Proceedings of the Apex Conference on Cloud Architecture (ACCA)*. Meridian Computer Society, 2025, pp. 112–125.