Building Cognitive Commerce Platforms in Go Architecting LLM-Driven Intelligence for Grocery Retail and E-Commerce

By Sanjay Basu · Global Head for AI / GenAI Products, TCS · Conf42 Golang 2026

The Shift: From Rule-Based to Cognitive

Grocery retail and e-commerce are crossing a fundamental threshold. Systems that once relied on deterministic rules and hand-crafted logic are giving way to **intent-aware, LLM-driven platforms** capable of reasoning, adapting, and responding in real time.

This is not a model-swapping exercise. It demands a wholesale rethink of backend architecture, orchestration patterns, and data pipelines and Go is exceptionally well-positioned to carry that weight.

What We'll Cover Today

01

Architecture Foundations

API gateways, streaming pipelines, and concurrency-safe LLM integration

02

Commerce Intelligence

Conversational search, personalisation, and product discovery

03

Merchandising Automation

Taxonomy generation, attribute extraction, and catalogue enrichment

04

Operational Intelligence

Demand sensing, substitution logic, and fulfilment orchestration

05

Agentic Systems

Controlled autonomy in warehouse, micro-fulfilment, and last-mile delivery

Why Go for Cognitive Commerce?

Concurrency at Scale

Goroutines and channels handle thousands of simultaneous LLM calls, stream tokens, and coordinate downstream services without thread-pool exhaustion.

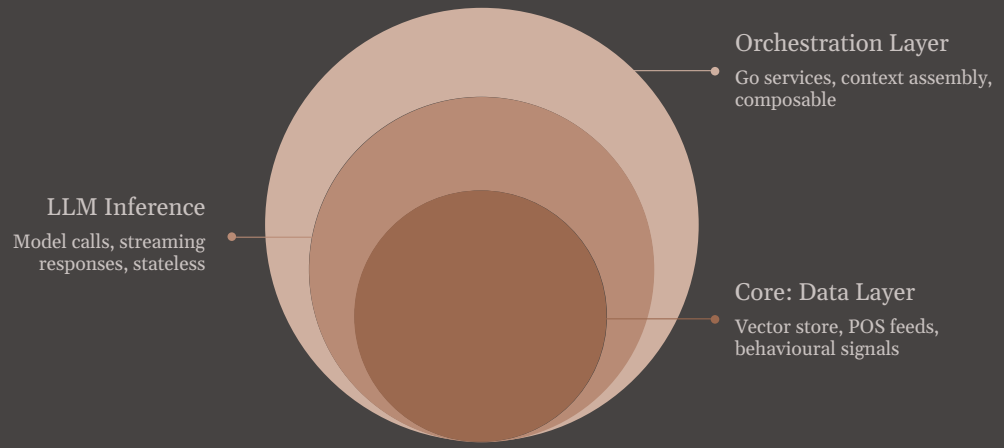
Predictable Latency

Go's low-pause GC and lean runtime deliver the sub-100ms tail latencies that real-time commerce demands even under surge load.

Operational Simplicity

Single static binaries, fast cold starts, and first-class observability tooling reduce operational overhead in high-churn microservice estates.

Architectural Backbone: Integrating LLMs into Commerce Stacks



Each layer is independently scalable. Go services own orchestration — keeping model calls stateless and composable.

Key Design Principles

Prompt Routing at the Gateway

Route by intent class before hitting inference to reduce cost and latency.

Context Assembly in Go

Aggregate user history, inventory state, and session context server-side before prompt construction.

Streaming-First Responses

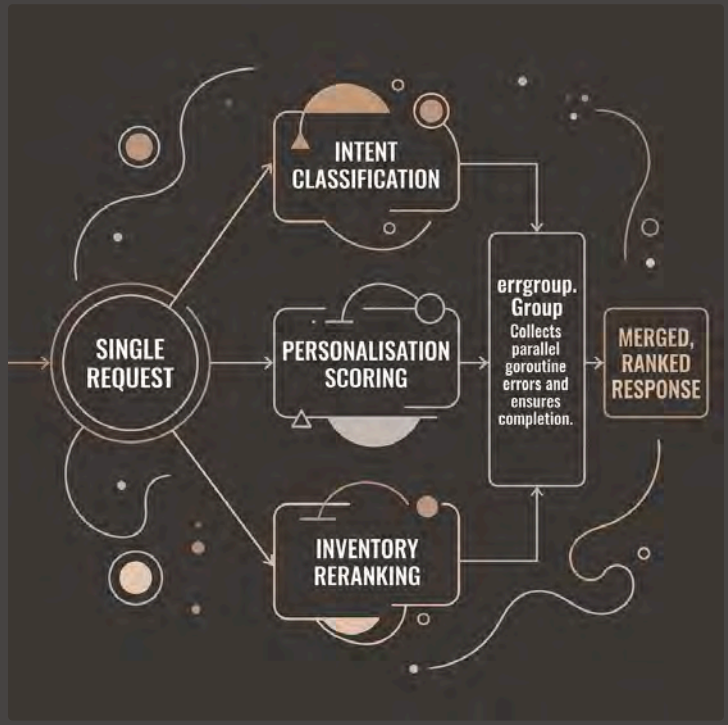
Use `io.Pipe` and Server-Sent Events to stream tokens directly to the client.

Concurrency Patterns for LLM Pipelines

The Challenge

A single commerce request may fan out to **multiple LLM calls** intent classification, personalisation scoring, inventory-aware reranking all within a single latency budget.

Go's `errgroup`, bounded worker pools, and context propagation make this tractable without external orchestration frameworks.



Commerce Intelligence: Search, Discovery & Personalisation

- ## Conversational Search

  LLMs parse natural-language queries ("something quick and high-protein for tonight") into structured filters, resolved against a vector index of product embeddings served from Go.

- ## Real-Time Personalisation

  Behavioural signals from clickstream, basket history, and loyalty data are assembled in Go and injected into the prompt context for session-scoped recommendations.

- ## Multimodal Product Discovery

  Image embeddings and text embeddings are unified in a shared vector space. Go coordinates the multimodal retrieval pipeline, merging ranked results before response.

Vector Search Infrastructure in Go

Production Considerations

- Run embedding generation as a background Go worker pool never block the request path
- Use `pgvector` for teams already on Postgres; graduate to Weaviate or Qdrant at scale
- Cache popular query embeddings with a short TTL to absorb search burst
- Gate ANN index refreshes behind a feature flag to control re-indexing cost

Merchandising Automation at Catalogue Scale

Taxonomy Generation

LLMs classify new SKUs into hierarchical category trees. Go services batch-process supplier feeds, deduplicate, and write enriched records asynchronously.

Attribute Extraction

Unstructured product descriptions, images, and spec sheets are parsed by multimodal models. Go pipelines normalise and validate extracted attributes against the catalogue schema.

Catalogue Enrichment

Missing fields nutritional data, allergen flags, sizing are inferred and populated at ingestion time, keeping the assortment machine-readable for downstream search and compliance.

Operational Intelligence: Demand Sensing & Substitution

Signal Sources

Go microservices ingest real-time signals across three domains and feed them into LLM-assisted decision services:

Online Behaviour

Clickstream, basket abandonment, search queries

Point-of-Sale

Transaction velocity, out-of-stock signals, shrinkage alerts

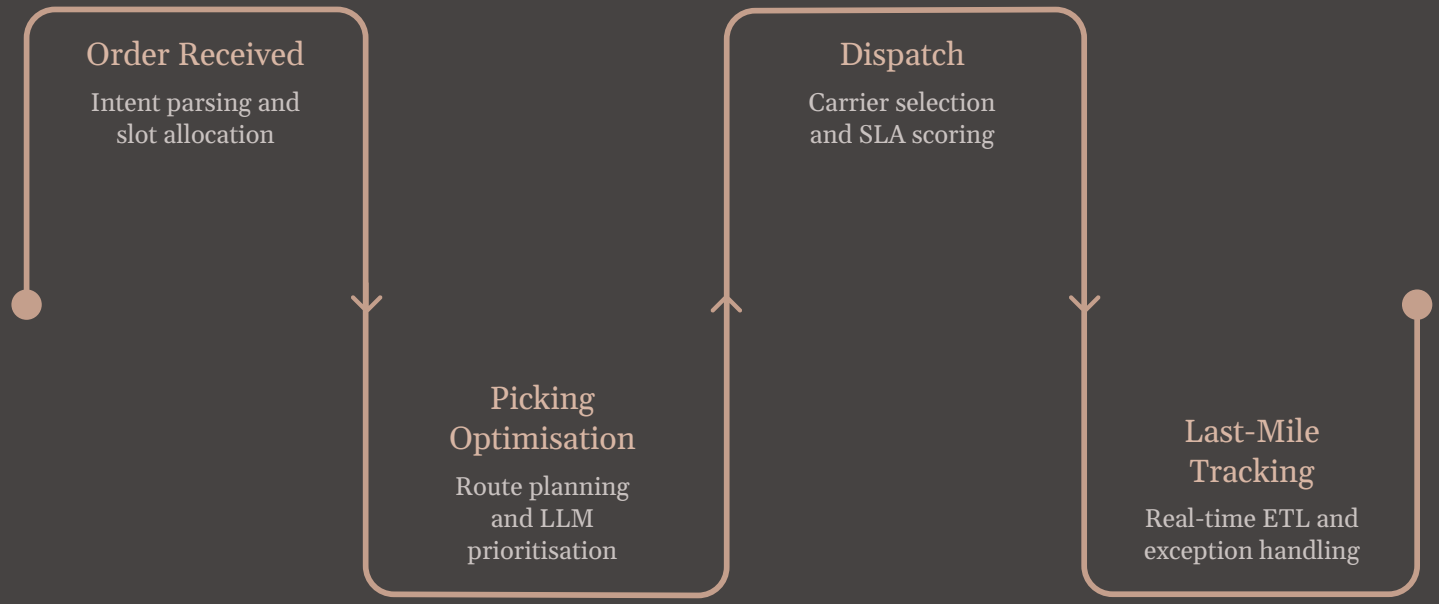
Logistics

Supplier lead times, warehouse stock levels, delivery SLAs

Substitution Intelligence

When a product is unavailable, an LLM-backed substitution service ranks alternatives by **semantic similarity, dietary compatibility, price delta, and margin** surfacing the best option in under 200ms.

Fulfilment Orchestration with Go Microservices



Go's channel-based coordination allows each stage to emit events consumed by the next, with dead-letter queues and circuit breakers preventing cascade failures across the fulfilment chain.

Agentic Architectures: Controlled Autonomy

- Warehouse Management

  LLM agents interpret pick exceptions, replenishment triggers, and slot conflicts Go services execute the resolved actions with full audit trails.

- Micro-Fulfilment

  Agents coordinate robot task queues, adapting to real-time inventory deltas. Go goroutines manage concurrency across hundreds of simultaneous pick tasks.

- Last-Mile Delivery

  Route optimisation, customer communication, and exception escalation are agent-orchestrated, with Go services maintaining state and enforcing SLA constraints.

Observability, Reliability & Cost Governance

Production Non-Negotiables

- **Trace every LLM call** end-to-end with OpenTelemetry correlate token cost to business outcome
- **Circuit-break aggressively** degrade to deterministic fallback rather than let latency spike
- **Tag requests by intent class** to route cheap queries to smaller, faster models
- **Set hard token budgets** per request type; enforce them in the Go gateway layer

Key Takeaways

- **Architecture First**

  LLM integration is an architectural challenge. Design for streaming, fan-out concurrency, and context assembly before choosing a model.

- **Go is a Natural Fit**

  Goroutines, low-latency runtime, and static binaries make Go the right substrate for high-throughput, real-time cognitive commerce services.

- **Agentic ≠ Uncontrolled**

  Autonomous agents require stronger observability and stricter circuit-breaking not less. Go services are the enforcement layer.

- **Govern Cost from Day One**

  Token budgets, model tiering, and per-intent routing are not afterthoughts build them into the gateway before your first production deployment.

Thank You!