# Building Real-Time AI Decision Systems with Large Language Models

How enterprises are combining LLMs, analytics, and modern data platforms to support scalable operational intelligence across complex environments.

Conf42 Large Language Models (LLMs) 2026

**By Mohanraj Varatharaj**
**Lead software engineer**

Session Overview

# From AI Experiments to Operational Intelligence

The challenge for modern enterprises is no longer simply deploying AI models it's integrating them into reliable, real-time decision systems that operate at scale. This session covers the architectural, operational, and governance dimensions of that shift.

01

### The State of Enterprise LLMs

Where organizations stand today and why static reporting falls short

02

### Architecture Patterns

Integrating LLMs, streaming data, and retrieval-based systems

03

### Implementation Challenges

Fragmented data, governance, scalability, and trust

04

### Production Reliability

Observability, performance, and operational best practices

# The New Mandate: Real-Time Decision Intelligence

Enterprises have spent years accumulating data and standing up isolated AI experiments. The competitive pressure has now shifted: organizations need AI systems that inform decisions *in the moment* not hours or days later via static reports.

### Then: Isolated AI

- Batch-processed analytics
- Siloed model deployments
- Manual handoff to decision-makers
- Weeks-long insight cycles

### Now: Operational Intelligence

- Continuously evolving AI systems
- Streaming data + LLM-assisted insights
- Automated, real-time recommendations
- Sub-second decision support

# Core Architectural Components

Reliable real-time AI decision systems are not monolithic they are composed of specialized layers that each handle a distinct concern. Getting the boundaries right between these layers is critical for maintainability and performance.

### Data Ingestion Layer

Streaming pipelines (Kafka, Kinesis, Flink) that ingest, normalize, and route events from fragmented enterprise sources in real time.

### LLM Reasoning Layer

Large language models augmented with retrieval-based context (RAG) that generate structured recommendations from live operational signals.

### Decision Engine

Orchestration logic that evaluates LLM outputs against business rules, confidence thresholds, and governance constraints before acting.

### Observability Layer

Logging, tracing, and drift-detection systems that maintain transparency into model behavior, latency, and decision quality over time.

# From Static Pipelines to Continuously Evolving Systems


Most enterprises enter this journey at the reporting stage. The leap to real-time LLM-integrated decision systems requires investment across infrastructure, data quality, and organizational readiness  not just model capability. Each stage unlocks new decision velocity while introducing new operational requirements.

# Fragmented Data: The Root Problem

Most enterprise data landscapes are the result of decades of acquisitions, system migrations, and organic tooling growth. LLMs operating over fragmented, inconsistent data sources will produce unreliable and potentially harmful recommendations.

### Schema Inconsistency

The same business entity (a customer, a transaction, a product SKU) often exists under different schemas across CRM, ERP, and data warehouse systems, requiring semantic normalization before any LLM can reason reliably over it.

### Latency Mismatches

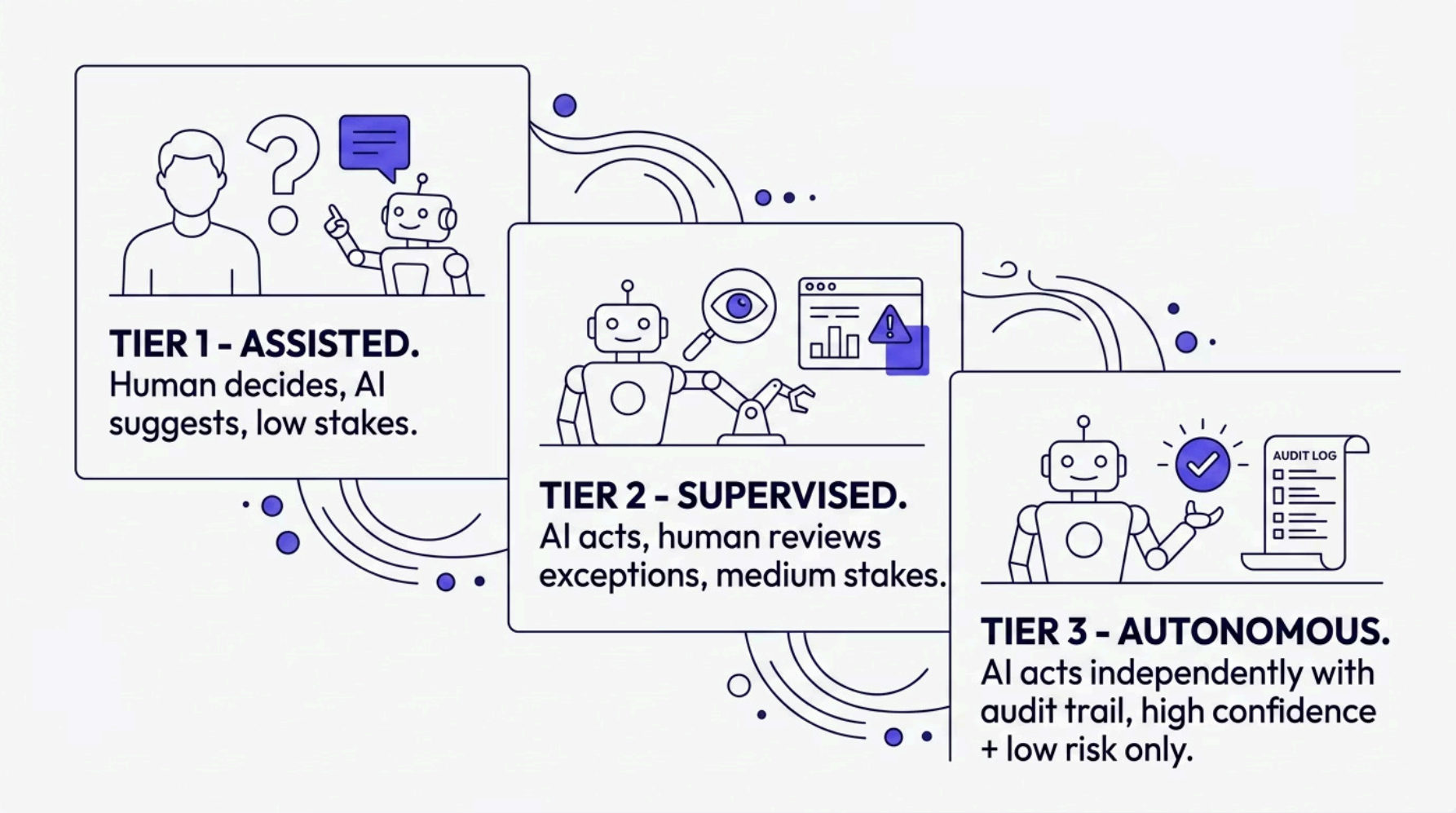
Mixing real-time event streams with batch-updated reference data introduces temporal inconsistencies. A decision model seeing yesterday's inventory against today's demand signal will generate systematically biased outputs.

### Ownership Gaps

In large organizations, data ownership is often unclear. No defined owner means no SLA for freshness, no accountability for quality, and no clear path to remediation when an LLM surfaces a bad recommendation.

# Governance, Trust, and the Human-in-the-Loop Question

As LLM-driven systems move from experimental to operational, governance becomes non-negotiable. Organizations must define how much autonomy AI systems are granted and under what conditions humans must remain in the decision loop.



# Infrastructure Scalability: Where Systems Break

LLM inference is expensive, stateful, and latency-sensitive. Scaling real-time decision systems requires deliberate infrastructure choices at every layer from model hosting to context management to downstream API contracts.

## Common Scalability Bottlenecks

- **Token throughput limits** LLM APIs impose rate limits that collapse under enterprise request volumes without careful batching and queuing strategies
- **Context window management** Retrieval-augmented systems must carefully control what context is injected to avoid latency spikes and cost overruns
- **Cold-start latency** Serverless LLM deployments introduce unpredictable warm-up delays incompatible with sub-100ms SLAs
- **Stateless vs. stateful trade-offs** Maintaining conversation or session context across distributed services significantly complicates horizontal scaling

## Design Principles for Scale

- Decouple inference from orchestration
- Cache deterministic LLM outputs aggressively
- Use async patterns for non-critical decisions
- Set hard latency budgets per decision type
- Build circuit breakers around LLM calls

# Retrieval-Augmented Generation in Operational Contexts

RAG architectures allow LLMs to reason over fresh, domain-specific knowledge without expensive fine-tuning cycles. In real-time decision systems, RAG is often the difference between a hallucinating model and a trustworthy one.

### Vector Store Design

Operational RAG systems require low-latency vector retrieval. Index design, chunking strategy, and embedding freshness all directly impact the quality of context delivered to the LLM at inference time.

### Retrieval Precision

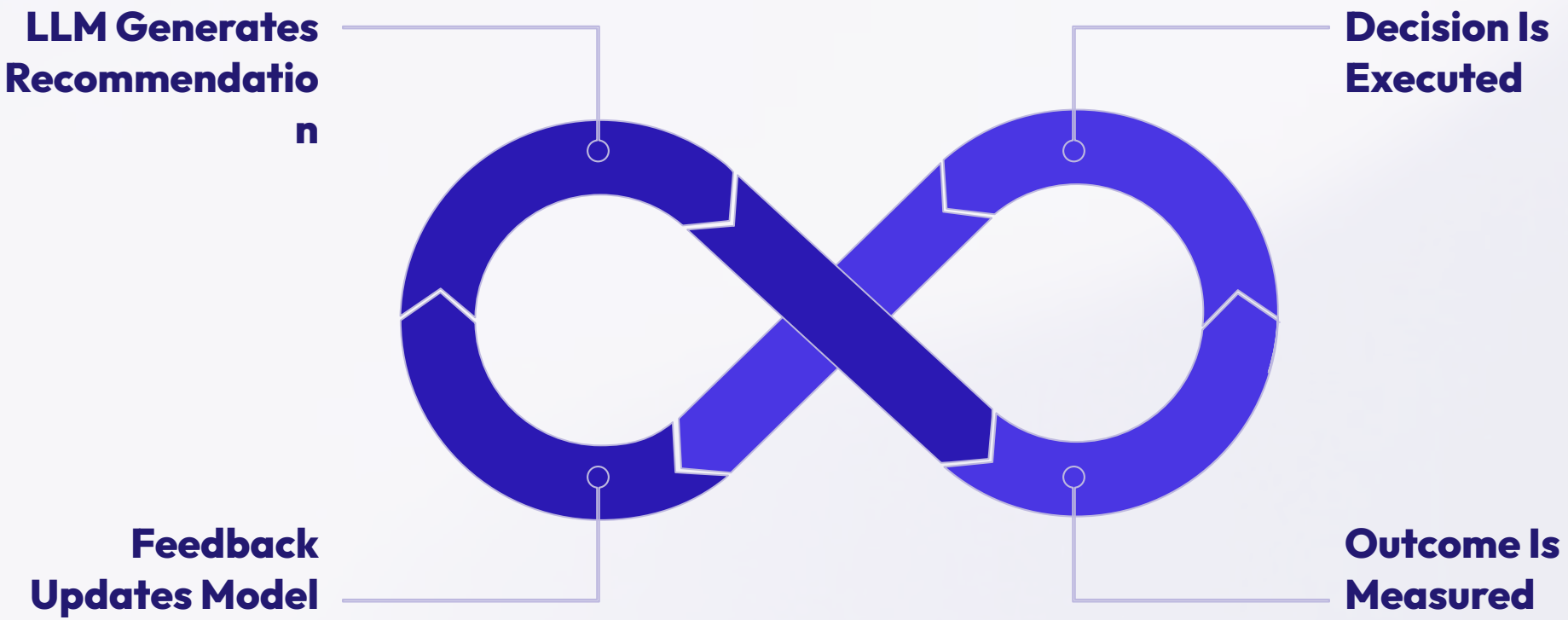
Over-retrieval floods the context window with noise; under-retrieval leaves the model operating on incomplete information. Hybrid retrieval (dense + sparse) combined with re-ranking substantially improves decision-relevant recall.

### Index Update Cadence

For operational decisions, stale retrieval indexes are as dangerous as stale models. Define update SLAs per knowledge domain and implement change-detection triggers to avoid serving outdated context.

# Reinforcement Learning for Continuous Decision Improvement

Reinforcement learning from human feedback (RLHF) and online RL techniques allow decision systems to improve continuously as they accumulate operational experience closing the loop between AI recommendations and measurable outcomes.




Implementing this loop in production requires reliable outcome telemetry, a feedback attribution model, and careful guardrails to prevent the system from over-fitting to short-term signals at the expense of long-term decision quality.

# Observability: The Non-Negotiable Requirement

In production AI systems, observability is not optional infrastructure it is a first-class architectural concern. Without deep visibility into model behavior, organizations cannot detect drift, debug failures, or justify AI-assisted decisions to regulators and stakeholders.

### Model Monitoring

Track output distributions, confidence scores, and token-level latency. Establish baselines at deployment and alert on statistically significant deviations.

### Decision Tracing

Every AI-assisted decision should be traceable back to its inputs, retrieved context, model version, and the business rules applied enabling root-cause analysis when decisions go wrong.

### Drift Detection

Concept drift in enterprise data is inevitable. Automated drift detection pipelines must trigger retraining or fallback routing before degraded model quality reaches end users.

### Explainability Logs

Regulatory and internal audit requirements demand that decision systems can explain their reasoning. Store structured rationale alongside each recommendation for downstream review.

# Fallback & Safe-Fail Mechanisms

Real-time AI systems will hit edge cases, latency spikes, and model failures. Resilient systems need explicit fallback behavior before it is ever needed.

## Deterministic Fallbacks

When an LLM times out or returns bad output, switch to a tested rules engine. Keep the response fast, predictable, and safe.

## Graceful Degradation

If the primary model slows down, route to a simpler baseline. That preserves service continuity without blocking users.

⚠ Silent failure is the bigger risk. Safe-fail must be designed in.

# Guardrails & Evaluation (Pre- and Post-Inference)

Observability tells you what happened. Guardrails prevent bad outcomes before they occur.

### Input Guardrails

Block prompt injections, toxic inputs, and out-of-scope queries before they reach core logic.

### Output Evaluation

Run schema checks, regex, or evaluator models to catch bad outputs before any decision is executed.

| Layer | Method | Purpose |
| --- | --- | --- |
| Input | Prompt injection detection | Security |
| Output | Schema / regex validation | Compliance |
| Output | Evaluator model scoring | Quality assurance |

# Cost & Latency Budgeting

In real-time production, cost and latency shape every architectural choice. Define both upfront.

### Semantic Caching

Reuse identical or near-identical responses to cut API spend. Best for frequent, repetitive decisions.

**~60%**

cacheable queries

**<200ms**

p95 latency target

### Hybrid Model Routing

Use small local models for fast, common decisions. Send complex edge cases to larger LLMs.

**10–100x**

cost gap

**3 tiers**

routing levels

# Key Takeaways & Next Steps

Building real-time AI decision systems is an engineering discipline, not just a model selection exercise. Success requires treating data quality, governance, and observability with the same rigor as model architecture.

### Audit your data foundations first

No LLM architecture compensates for fragmented, low-quality, or poorly governed data. Data readiness is a prerequisite, not an afterthought.

### Define decision tiers before you build

Classify each decision type by stakes, reversibility, and required latency. This taxonomy drives every downstream architectural choice.

### Build observability from day one

Retrofitting observability into a production AI system is significantly more costly than building it in from the start. Instrument everything at launch.

### Close the feedback loop intentionally

Systems that improve through operational experience outperform those that are statically deployed. Invest in outcome measurement infrastructure early.

# Thank You!