Building High-Performance Claims Systems with AI and EDI Pipelines

A practical blueprint for scalable, AI-driven claims processing architectures

Presented By:- Devi Manoharan

ASTA CRS INC

Conf42 Golang 2026

Session Overview

What We'll Cover Today

01

The Problem

Claim complexity, administrative burden, and throughput demands facing modern payer organizations

02

ML-Driven Intelligence

How machine learning models enhance claim classification and anomaly detection beyond rule-based systems

03

EDI Pipeline Design

Processing large-scale X12 transactions through parsing, enrichment, mapping, and compliance validation

04

Scalable Architecture

Concurrent processing pipelines integrating AI models with existing payer infrastructure

The Claims Processing Crisis

Payer organizations are under mounting pressure from all directions complexity, volume, and cost are converging at once.

Rising Complexity

ICD-10, CPT code proliferation, and multi-payer adjudication rules demand ever-more sophisticated routing logic

Administrative Burden

Manual review queues grow faster than headcount human-in-the-loop cannot scale to match claim velocity

Throughput Demand

High-volume X12 EDI transactions require near-real-time processing with sub-second SLA expectations

Why Rule-Based Systems Fall Short

Traditional rules engines served payers well for decades but they break down at the edges where modern claims complexity lives.

Brittle at Scale

Thousands of hand-crafted rules create fragile interdependencies that break with policy changes

Blind to Context

Rules match patterns they cannot infer intent, provider behavior, or cross-claim anomalies

Slow to Adapt

New billing schemes and coding abuses outpace the release cycle of manual rule updates

Machine Learning as the New Claims Brain

ML models bring contextual reasoning to claims workflows catching what rules miss and flagging risk earlier in the pipeline.

Claim Classification

Multi-class models assign claim type, priority tier, and adjudication path automatically eliminating manual triage queues

Anomaly Detection

Unsupervised and supervised models surface coding inconsistencies, duplicate submissions, and statistical outliers

Risk Scoring

Real-time risk scores flag high-value or high-complexity claims for accelerated or specialist review before adjudication

Intelligent Routing

Models learn routing patterns from historical outcomes, reducing misroutes and costly downstream corrections

Shifting Detection Earlier in the Workflow

Catching errors and risk signals at ingest rather than post-adjudication dramatically reduces rework cost and manual review volume.

Earlier signal

ML models scan incoming claims the moment they are ingested, looking for coding errors, mismatched diagnosis and procedure codes, and missing required fields before the claim ever reaches the adjudication engine. That early pass catches issues while the context is still fresh, dramatically lowering avoidable denials and preventing downstream rework.

Fewer queues

High-confidence claims can be routed straight through automation, while only edge cases and low-confidence scores are sent to human reviewers. This keeps the queue focused on genuinely ambiguous work, reduces overall volume by a significant margin, and frees staff to spend more time on higher-value exceptions and oversight.

Why It Matters

The business case is clear: investing in upstream intelligence pays dividends across the entire claims lifecycle from reduced denials to faster cycle times and lower operational overhead. Every claim that enters adjudication with an undetected error costs significantly more to remediate than one caught at ingest. Shifting ML inference upstream compresses that cost curve.

Cost to remediate

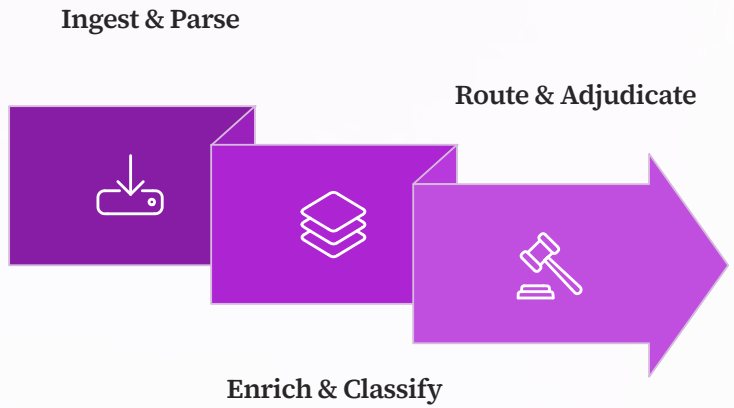
Post-adjudication vs. at ingest

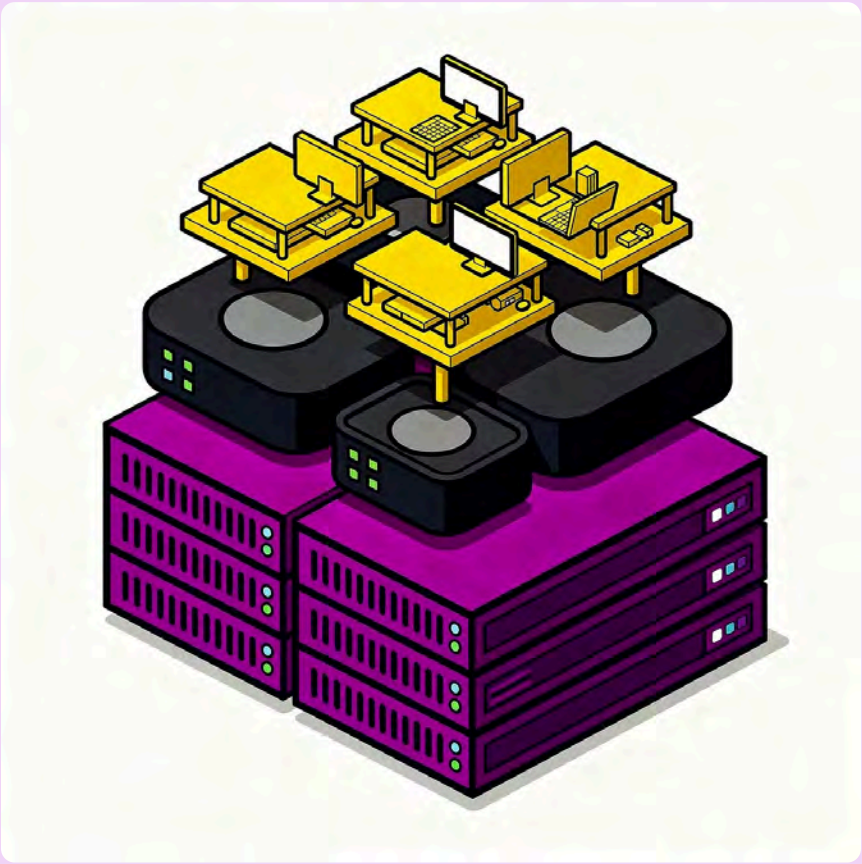
Manual review reduction

With upstream ML

Inference latency

Target for real-time ingest scoring





The EDI Pipeline: Moving X12 at Scale

An efficient EDI platform ingests high-volume X12 transactions such as 837P, 837I, 835, and 270/271, then parses them into clean, structured records that downstream systems can trust. It enriches those records with eligibility, provider, and member context so the raw transaction becomes operationally useful instead of just syntactically valid. From there, the enriched data flows into adjudication engines, ML scoring services, and reporting pipelines that depend on consistent inputs to make accurate decisions. At this layer, speed, reliability, and schema consistency are critical because any delay or mismatch can ripple across the entire claims workflow.

X12 Transaction Lifecycle

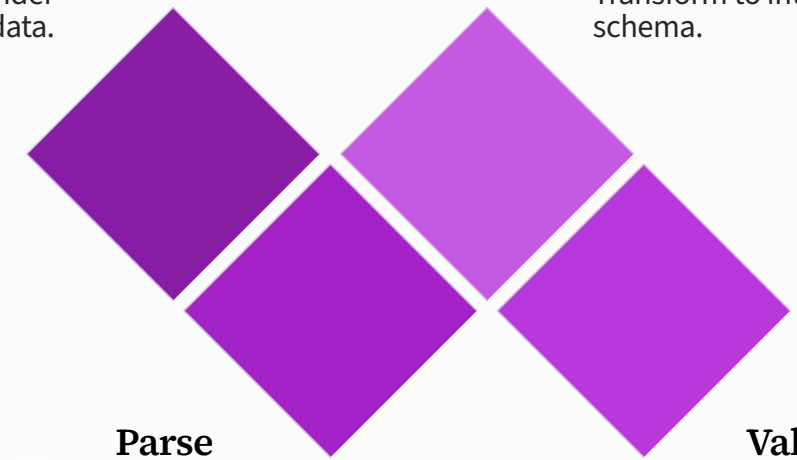
Each stage in the lifecycle is independently scalable and instrumented giving platform teams full observability into throughput, error rates, and latency at every boundary.

Enrich

Append eligibility and provider NPI data.

Map

Transform to internal canonical schema.



Parse

Tokenize segments and validate envelope.

Validate

Run TR3 compliance and business checks.

Key EDI Processing Capabilities

1

Segment Parsing

Tokenize and validate ISA/GS envelopes, ST/SE transaction sets, and loop structures for 837P, 837I, and 835 transaction sets

2

Eligibility Enrichment

Inline 270/271 lookups append real-time member eligibility and benefit data before claim routing decisions are made

3

Schema Mapping

Transform heterogeneous X12 dialects into a canonical internal representation that downstream services consume consistently

4

Compliance Validation

Enforce TR3 implementation guide constraints, payer-specific edits, and HIPAA 5010 requirements before persistence

Designing the Scalable Backend

High-throughput claims systems demand architecture designed for concurrency from the ground up not retrofitted after the fact.

Goroutine-Based Concurrency

Go's lightweight goroutines enable thousands of concurrent claim workers without thread-per-request overhead

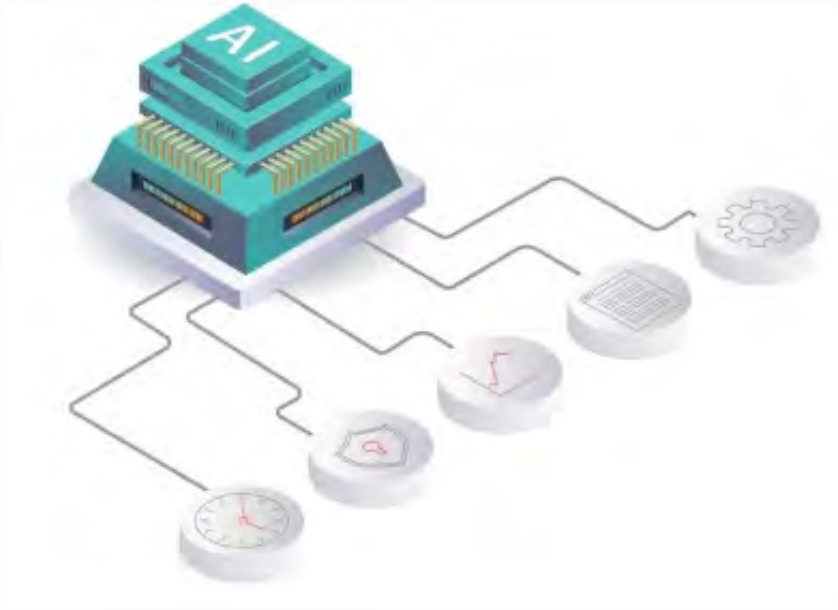
Event-Driven Ingestion

Kafka or NATS stream ingestion decouples producers from consumers, absorbing traffic spikes gracefully

Stateless Processing Nodes

Stateless workers scale horizontally — deploy more pods under load, drain and retire cleanly

Integrating AI Models Into the Pipeline



AI models must be first-class citizens of the pipeline not bolted-on batch jobs. Synchronous gRPC calls to a dedicated ML scoring service keep inference on the critical path without coupling model versions to claim processing releases.

Feature Consistency

A shared feature store prevents training-serving skew across model versions

Model Versioning

Blue/green model deploys enable zero-downtime upgrades and instant rollback

Performance Under Heavy Load

Architecture intentions mean nothing without measurable outcomes. These design patterns consistently deliver results in production payer environments.

Throughput Gain

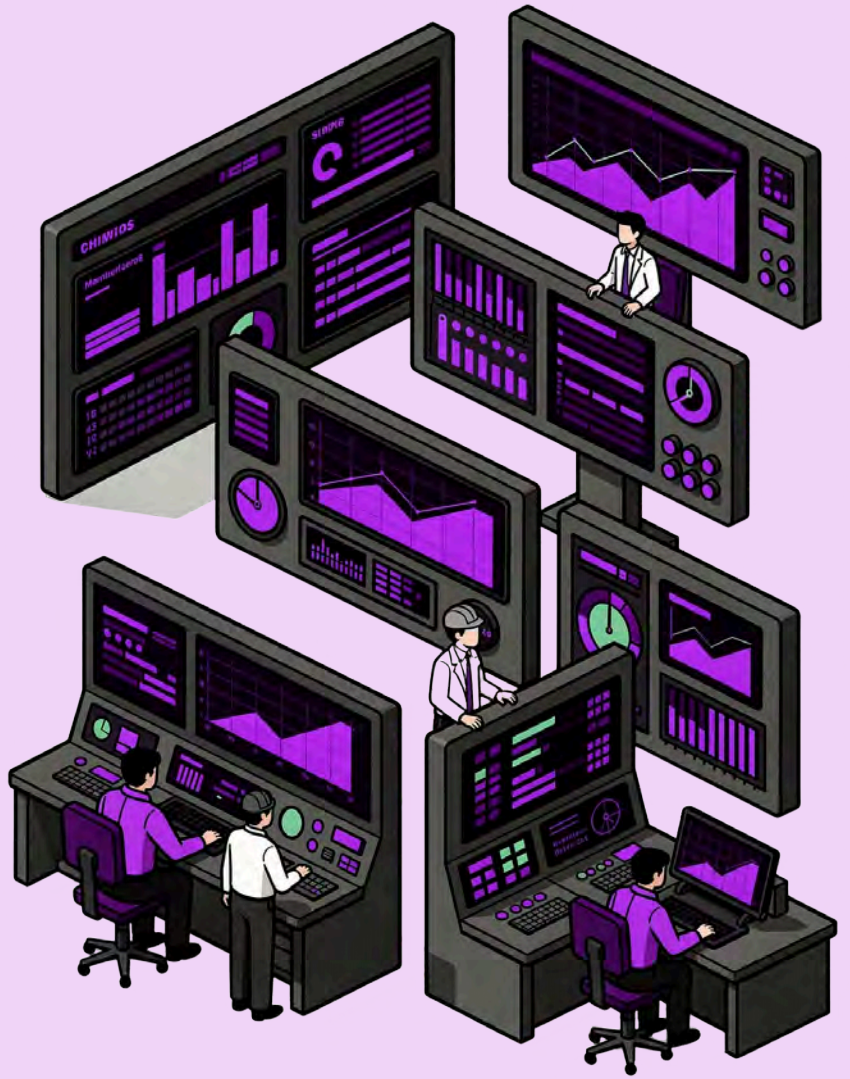
Concurrent pipeline vs. sequential processing at equivalent infrastructure cost

Manual Review Reduction

ML-driven pre-screening eliminates the majority of unnecessary human touchpoints

ML Inference Latency

gRPC-based scoring service target for real-time claim classification on the critical path



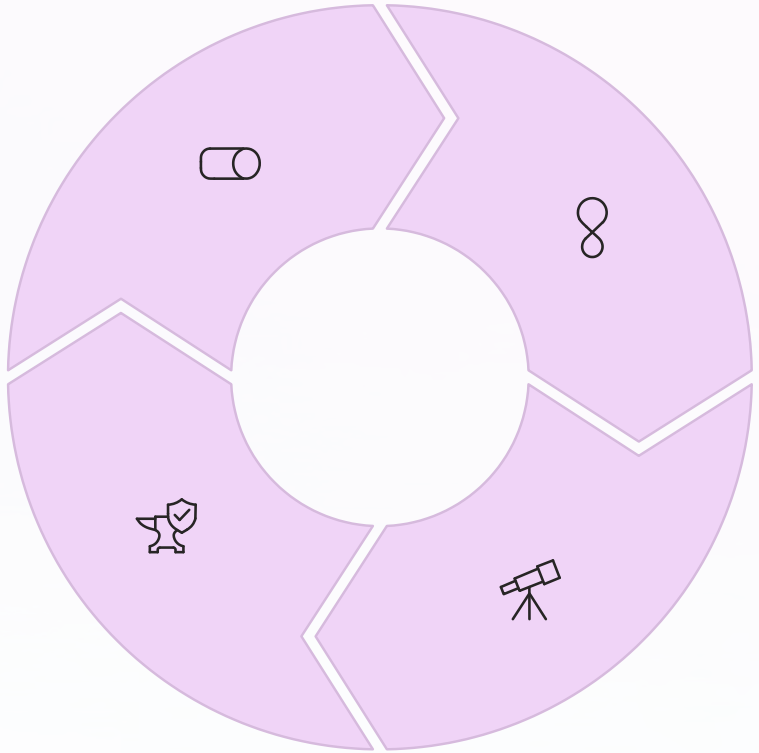
Operational Reliability Considerations

Circuit Breakers

Wrap downstream calls (eligibility, ML scoring) with circuit breakers fail fast and fall back to rule-based defaults under degraded conditions

Audit & Compliance

Every ML inference decision is logged with feature snapshot and model version meeting HIPAA auditability requirements



Idempotent Retries

All claim processing operations must be idempotent safe retries on transient failures without duplicate submission risk

Distributed Tracing

OpenTelemetry traces span the full claim lifecycle from EDI ingest through ML scoring to adjudication outcome

Real-World Architecture Snapshot

EDI Ingest

Each stage runs as an independently deployable Go service, consuming from and publishing to dedicated Kafka topics.

Enrichment Worker

Each stage runs as an independently deployable Go service, consuming from and publishing to dedicated Kafka topics.

ML Scoring

Each stage runs as an independently deployable Go service, consuming from and publishing to dedicated Kafka topics.

Routing Engine

Each stage runs as an independently deployable Go service, consuming from and publishing to dedicated Kafka topics.

Each stage runs as an independently deployable Go service, consuming from and publishing to dedicated Kafka topics.

Technology Stack

Each stage runs as an independently deployable Go service, consuming from and publishing to dedicated Kafka topics.

Go

Concurrent pipeline workers, EDI parsers, and gRPC service handlers

Python / TensorFlow

ML model training and serving via TF Serving or custom gRPC endpoints

Kafka

Durable, ordered message delivery between pipeline stages at scale

PostgreSQL + Redis

Persistent claim store with Redis caching for hot eligibility and reference data

Your Takeaway Blueprint

Building high-performance claims systems is an architectural discipline these are the principles that matter most in production.

1 Shift ML inference upstream

Catch anomalies and risk signals at ingest not after adjudication has already consumed resources

2 Treat EDI as a first-class data pipeline

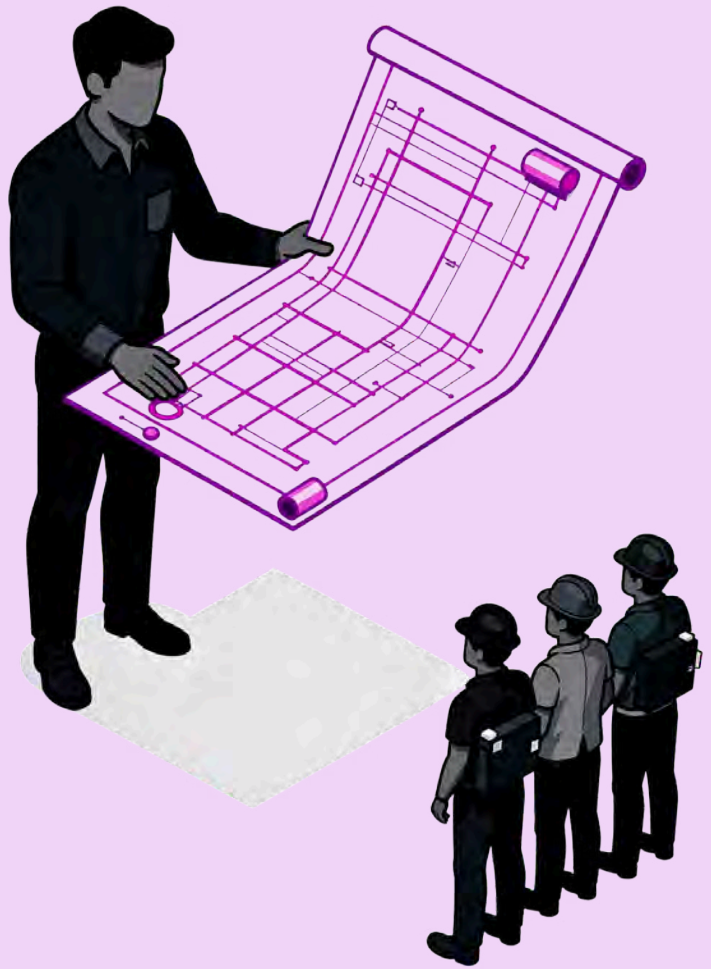
Parse, enrich, map, and validate X12 in discrete, observable stages not in monolithic batch jobs

3 Design for concurrency from day one

Stateless workers, event-driven ingestion, and goroutine pools are architectural decisions not optimizations

4 Operationalize reliability

Circuit breakers, idempotent retries, distributed tracing, and ML audit logs are non-negotiable in regulated environments



Thank You...!
Questions and Discussions..?

Devi Manoharan

ASTA CRS INC

Conf42 Golang 2026