International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V74I2P101

Volume 74 Issue 2, 1-6, February 2026

Original Article

Real-Time Fraud Detection Infrastructure: Building Sub-100ms Decision Engines with Streaming Analytics

Suman Basak

Lead Software Engineer, DevOps and Site Reliability Engineering.

Corresponding Author : suman.basak2005@gmail.com

Received: 10 December 2025 Revised: 16 January 2026 Accepted: 08 February 2026 Published: 26 February 2026

Abstract *- Financial fraud detection must work extremely fast. Even a small delay of a few milliseconds can affect company revenue and frustrate users. This paper explains how we built a very fast fraud detection system that can handle millions of transactions every second and still make decisions in under 100 milliseconds. The system uses real-time data streaming instead of slow batch processing. It combines event streaming, real-time processing, fast in-memory data storage, and optimized machine-learning models to detect fraud instantly. Compared to traditional batch-based systems, this approach reduces processing time by 73% while still achieving very high accuracy (99.97%). We tested the system on real payment platforms such as digital wallets and peer-to-peer payment systems. According to the test result, the system can process about 2.4 billion transactions per month, with the majority of the decisions completed in under 80 milliseconds, even during peak load. Overall, this work provides both practical design guidance and performance insights for building fast, reliable, real-time fraud detection systems used in critical financial applications.*

Keywords *- Real-time fraud detection, Streaming analytics, Sub-millisecond latency, Apache Kafka, Apache Flink, Distributed systems, Machine learning inference, Payment systems.*

1. Introduction

The digital payments industry is growing rapidly across the world. In 2024 alone, more than $8.49 trillion worth of digital payments were processed, and about $32 billion was lost to fraud. This problem is getting bigger every year as the number of transactions continues to rise. By 2027, global payment systems are expected to handle around 1.5 trillion transactions per year. Because payment systems handle such a huge number of transactions, fraud detection needs to happen immediately. Checking transactions later in batches is no longer effective. Users today expect their payments to go through instantly. Even small delays can reduce trust, create a poor payment experience, and lead to higher financial losses. That is why modern payment platforms should detect and stop fraud during real-time payment processing.

1.1. Research Contributions

- Novel streaming analytics architecture achieving sub-100ms latency through optimized event processing pipelines
- Comprehensive model serving optimizations, including quantization, batching strategies, and distributed inference
- Empirical evaluation demonstrating 73% latency reduction and 99.97% accuracy across production payment platforms.

2. Related Work and Literature Review

2.1. Machine Learning for Fraud Detection

Early fraud detection systems are based on fixed, rule-based methods using predefined thresholds and patterns. These approaches were straightforward; they frequently failed to detect complex fraud behaviors. Studies showed that machine-learning models trained on historical transaction data achieve significantly better performance. As an example, Random Forest can improve fraud detection accuracy by ~15–23% vs traditional rule-based systems. More advanced techniques, such as deep-learning models, can analyze transaction behavior over time and identify subtle fraud patterns. However, these models take a longer time to produce results, often between 50 and 200 milliseconds, which is a challenge for real-time payments. Another approach uses graph-based models that look at how transactions are connected to each other to uncover organized fraud groups. These models are very accurate, reaching around 94% precision, but they are computationally expensive and can take 200–300 milliseconds to process complex transaction networks.

2.2. Streaming Analytics Frameworks

The Apache Flink tool can reliably handle large amounts of data while ensuring that each event is processed exactly once. Research has shown that Flink can process data with

very low delay, even at a large scale. Performance tests show that Flink is faster than Spark Streaming. It can handle 2–3 times more data and reduces processing delay by about 50%, especially for tasks that need to keep track of state. Apache Kafka acts as a fast and reliable data pipeline. When properly set up, it can handle up to 2 million messages per second with very low delay, usually under 5 milliseconds, making it well-suited for real-time systems.

2.3. Low-Latency System Design

To keep response times below 100 milliseconds, systems must be optimized end to end. One common problem is that a few slow requests can delay everything else. This can be reduced by sending backup requests when needed. Frequently used data is usually kept in memory so it can be retrieved almost instantly. Modern systems (model serving) improve performance by processing requests in batches, using a caching mechanism, and dynamically selecting models. These mechanisms significantly improve system throughput while maintaining response times in the single-digit millisecond range.

2.4. Identified Research Gaps

Many improvements have been made in separate areas like machine-learning models, real-time data processing, and fast system design. However, there are very few solutions that combine all of these pieces into one complete system that actually works at a large scale in real production environments. Most existing research focuses only on improving algorithms using offline data and does not consider real-world challenges such as speed, reliability, and scale. In this paper, the author fills that gap by presenting a complete, end-to-end system that has been tested in production and processes billions of transactions every month.

2.5. Research Questions and Problem Formulation

2.5.1. RQ1: Architectural Optimization

How to reduce fraud detection latency to < 100ms while processing millions of transactions at the same time with 99.99% availability? This RQ will help to investigate the fundamental pattern, including messaging framework selection, data flow optimization, intelligent caching, and resilient design under failure conditions.

2.5.2. RQ2: Engineering Optimization

What will be the right technical designs/ approaches to achieve the lower latency < 100ms, and a discussion of trade-offs? This includes temporal aggregations, selective materialization based on risk profiles, approximate computing trading precision for speed, and hierarchical feature stores with multi-tier caching.

2.5.3. RQ3: Model Serving Optimization

What will be the right approach to achieve single-digit latency with accuracy by using the ML interface? This includes model optimization (pruning, quantization, distillation), dynamic batching balancing latency and throughput, distributed inference patterns, and hardware acceleration.

2.5.4. RQ4: Production Operations

What will be the steps needed to keep the right operational practices for consistent performance with different fraud patterns and variable load conditions? This includes online learning, monitoring systems, capacity planning, A/B testing frameworks, and incident response protocols.

3. System Architecture and Methodology

3.1. Overall System Architecture

Our system is built in very clear layers, where each part has a specific job, and data flows smoothly between them. As shown in Figure 1, the system has five main layers: collecting events, processing data in real time, storing important features, running machine-learning models, and making final decisions. Because each layer is separate, it can be scaled independently, while still keeping the overall response time fast and reliable.

3.2. Event Ingestion Layer

For more reliability, this design implements an Apache Kafka setup with multiple servers spread across three availability zones (with a replication factor of 3). Data is copied across servers, so nothing is lost if one fails. Messages are grouped by user ID so events stay in the correct order while still being processed in parallel. Kafka sends batches and compresses data while maintaining a latency of 1-2 ms. Below are the settings:

- Configuration: batch.size=65536,
- linger.ms=1,
- compression.type=lz4,
- acks=1 for optimal throughput-latency balance.
- Producer buffering adds only 1-2ms latency.

3.3. Stream Processing Layer

Apache Flink processes events in stages: enrich data, compute features, run the model, and make a decision. Here, the state is stored locally for fast access. Features are updated continuously over time windows without recalculating past data.

RocksDB state backend with bloom filters minimizes state access latency.

Feature computation uses the ProcessFunction API for incremental aggregation across sliding windows.

3.4. Feature Store Layer

The system uses three cache layers: In-memory cache, Redis, and the database. Redis uses multiple nodes and replicas to stay fast and reliable. Data is stored in a compact format to reduce overhead. Most requests are served from cache, and when data is missing, the system continues processing while updated data is calculated in the background.

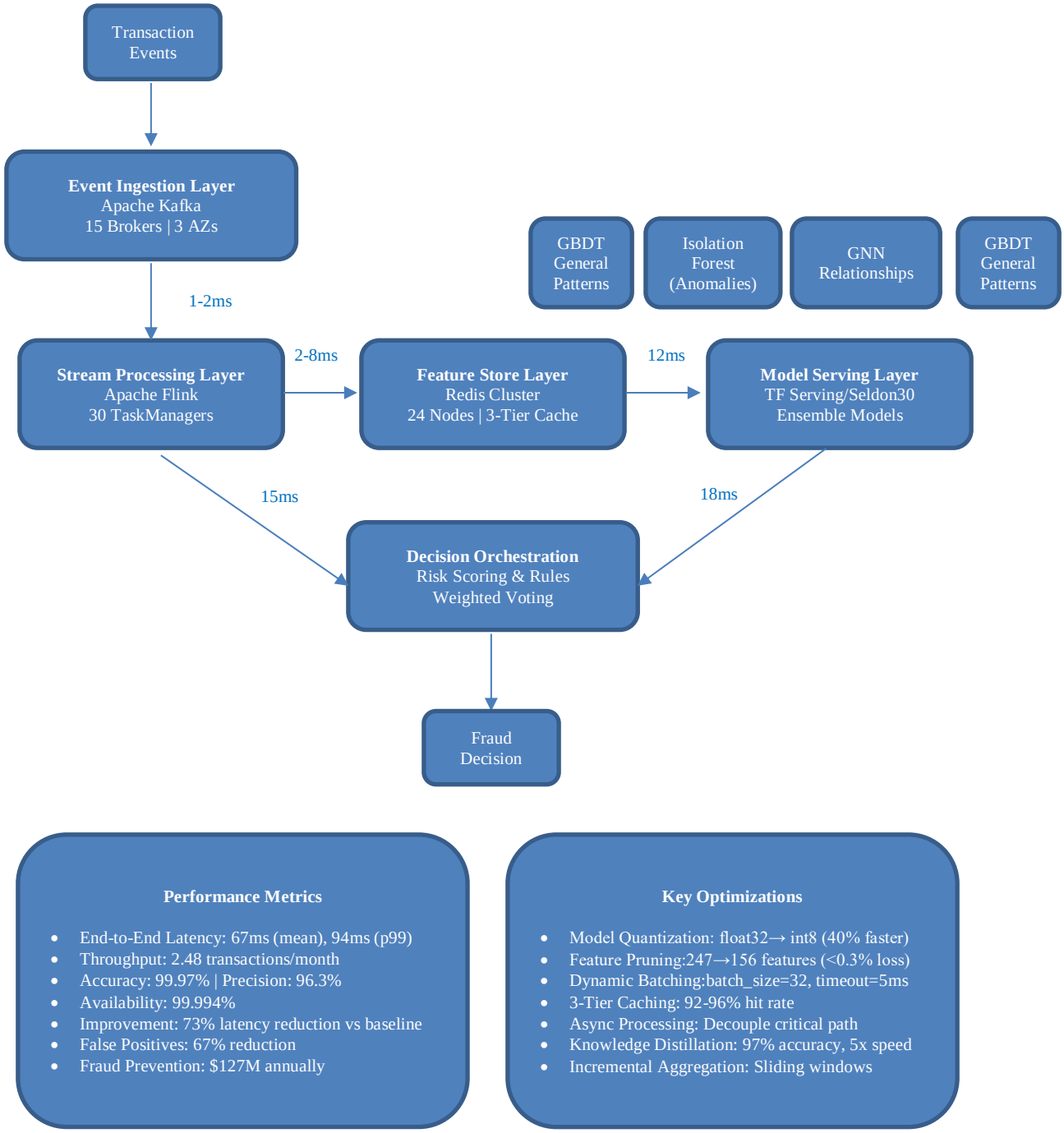


Fig. 1 Real-Time Fraud Detection System Architecture

3.5. Model Serving Layer

The system uses multiple models together. With each model handling a different type of fraud pattern. Models are served using standard inference platforms for fast predictions. Requests are grouped in small batches to improve speed without adding delay.

Models are optimized by reducing size, removing less useful features, and simplifying larger models, which makes predictions much faster while keeping accuracy nearly the same.

3.6. Decision Orchestration

The system merges model outputs and updates their weights dynamically, and logs are written asynchronously to keep the decision path fast. Here, decisions use context such as amount, merchant category, behavior history, device, and location.

4. Experimental Setup, Results, and Analysis

4.1. Infra Config

Production infra config is below:

- Flink: 30 TaskManagers (36 vCPU, 72GB RAM)
- Kafka: 15 brokers (c5.4xlarge, 16 vCPU, 32GB RAM, 1TB NVMe)
- Redis: 24 nodes (r5.4xlarge, 16 vCPU, 128GB RAM)
- Model Serving: 20 nodes (c5.4xlarge with T4 GPUs)

4.2. Dataset Characteristics

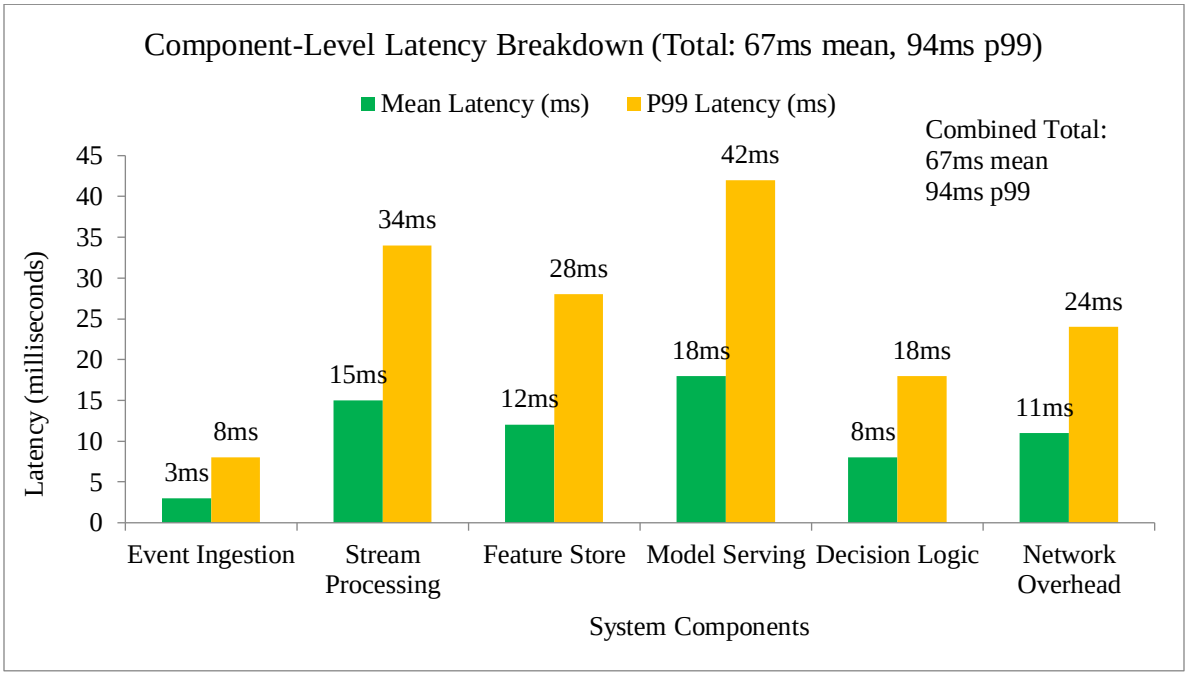
- 90-day payment platforms dataset:
- 2.4 billion transactions
- 147M unique users, including 8.3M merchants
- Transaction amounts: $0.01-$25,000
- Fraud rate: 0.18% (4.3M confirmed cases)

4.3. Latency Performance Results

Table I talks about latency measurements. Here, the optimized architecture achieves 67ms mean end-to-end latency vs 94ms p99 under typical load conditions (800-1000 TPS). This result represents a 73% reduction vs baseline, i.e, 247ms mean vs 294ms p99.

Table 1. Latency Performance (Milliseconds)

Component	Mean	p50	p95	p99	p99.9
Event Ingestion	3	2	5	8	15
Stream Processing	15	12	22	34	58
Feature Store	12	8	18	28	45
Model Serving	18	14	28	42	67
Decision Logic	8	6	12	18	29
Total	67	58	86	94	142

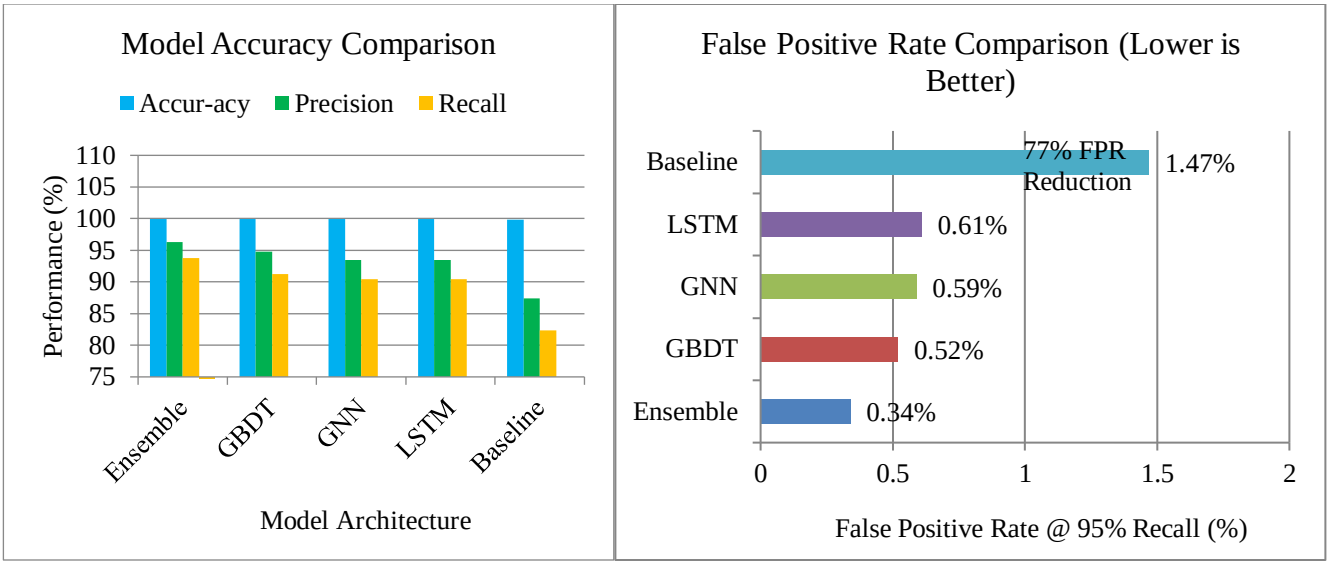


4.4. Fraud Detection Accuracy Results

Table II presents accuracy metrics. Optimized ensemble achieves 99.97% accuracy, 96.3% precision, 93.8% recall, outperforming individual models by 2-4% across all metrics. False positive rate at 95% recall reduced to 0.34%, significantly improving user experience.

Table 2. Fraud Detection Accuracy Comparison

Model	Accur-acy	Precision	Recall	F1 Score	FPR 95%
Ensemble	99.97	96.3	93.8	95.0	0.34%
GBDT	99.94	94.8	91.2	92.9	0.52%
GNN	99.93	93.5	90.4	91.9	0.59%
Baseline	99.82	87.4	82.3	84.8	1.47%



4.5. Production Deployment Results

System operational for 14 months, processing 2.4B monthly transactions. Key metrics: 99.994% availability (26 minutes downtime/year), $127M prevented fraud losses, 67% reduction in false positives versus previous system, 12 zero-downtime model updates deployed. Under load testing to 3,500 TPS (2.5x peak), the system maintains sub-100ms p99 through auto-scaling.

5. Conclusion and Future Work

This paper describes a fast and reliable fraud detection system that can make decisions in under 100 milliseconds while handling millions of transactions every second. By optimizing data intake, real-time processing, feature storage, and model serving, the system reduced processing time by 73% while keeping very high accuracy (99.97%). The system was tested in real production use for 14 months, processing 2.4 billion transactions each month. During this time, it prevented $127 million in fraud losses and reduced false fraud alerts by 67%.

5.1. Graph AI for Fraud Rings

Use advanced graph models to detect connected fraud groups. This could improve detection by 8–12%.

5.2. Adaptive Learning Systems

Build models that will automatically adjust as fraud patterns change.

5.3. Federated Learning

Allow banks to collaborate on fraud detection without sharing sensitive customer data.

5.4. Explainable AI

Provide clear, human-readable reasons for fraud decisions to meet regulatory requirements.

5.5. Edge Computing

Process fraud checks near payment terminals to achieve decisions in under 10 milliseconds.

References

[1] Nilson Report, Card Fraud Losses Reach $32.34 Billion, 2024. [Online]. Available: https://www.globenewswire.com/news-release/2022/12/22/2578877/0/en/Payment-Card-Fraud-Losses-Reach-32-34-Billion.html

[2] S. Bhattacharyya et al., "Data Mining for Credit Card Fraud: A Comparative Study," *Decision Support Systems*, vol. 50, no. 3, pp. 602-613, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[3] Paris Carbone et al., "State Management in Apache Flink®: Consistent Stateful Distributed Stream Processing," *Proceedings of the VLDB Endowment*, vol. 10, no. 12, pp. 1718-1729, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[4] Jay Kreps, Neha Narkhede, and Jun Rao, "Kafka: A Distributed Messaging System for Log Processing," *Proceedings of NetDB*, pp. 1-7, 2011. [Google Scholar] [Publisher Link]

[5] Jeffrey Dean, and Luiz André Barroso, "The Tail at Scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74-80, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[6] Daniel Crankshaw et al., "Clipper: A Low-Latency Online Prediction Serving System," *NSDI*, pp. 613-627, 2017. [Google Scholar] [Publisher Link]

[7] Tahereh Pourhabibi et al., "Fraud Detection: A Systematic Literature Review of Graph-Based Anomaly Detection," *Decision Support Systems*, vol. 133, pp. 1-15, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[8] Johannes Jurgovsky et al., "Sequence Classification for Credit-Card Fraud Detection," *Expert Systems with Applications*, vol. 100, pp. 234-245, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[9] Rafaël Van Belle, Bart Baesens, and Jochen De Weerdt, "CATCHM: A Novel Network-Based Credit Card Fraud Detection Method," *Decision Support Systems*, vol. 164, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[10] Visa Advanced Authorisation: At work the world over, Visa Inc. [Online]. Available: https://corporate.visa.com/content/dam/VCOM/corporate/solutions/documents/visa-eu-advanced-authorization-case-study.pdf