Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

From Streams to Security: Architecting a Production-Ready Fraud Pipeline with Flink and Kafka

Dhara Chimanbhai Trambadiya
Assistant Professor ,
Computer Engineering,
Noble University, Junagadh, Gujarat, India

Abstract : **As digital commerce increasingly requires instantaneous verification, conventional batch-oriented fraud identification frameworks are failing to meet modern latency requirements. This study proposes an advanced event-driven paradigm that integrates Apache Kafka, ksqlDB, and Apache Flink to facilitate real-time security analytics. By synthesizing diverse data streams—including geospatial coordinates, authentication logs, and transactional metadata—the system employs stateful stream processing to flag behavioral anomalies like geographical impossibilities and high-frequency velocity infractions. Our containerized deployment demonstrates a 94.2% sensitivity rate with sub-second response times. By merging deterministic rule sets with heuristic machine learning models, this research illustrates a transition from retrospective auditing to an autonomous, proactive defense layer within the enterprise data fabric.**

General Terms

Algorithms, Data Streaming, Machine Learning, Real-Time Systems, Pattern Recognition, Security, Distributed Systems, Event-Driven Architecture, Anomaly Detection, Big Data Processing.

Keywords - Real-time Fraud Detection, Apache Kafka, ksqlDB, Apache Flink, Stream Processing.

1.INTRODUCTION

In today's digital-first economy, real-time transaction systems serve as the backbone of financial services, e-commerce, and online platforms. However, this prevalence is accompanied by a concerning increase in fraudulent activities that take advantage of system latencies, protocol vulnerabilities, and increasingly sophisticated adversaries. Traditional fraud detection systems, typically batch-based or reliant on static rule sets, are fundamentally reactive, analyzing data hours or even days after fraudulent activities have occurred. This delay allows attackers to carry out high-speed, low-value transactions or mimic legitimate behaviors across distributed platforms, effectively avoiding outdated detection mechanisms. Recent studies indicate a significant uptick in digital fraud. The Reserve Bank of India reported a 216% increase in transaction volume and a 10% rise in value from 2019 to 2022, considerably expanding the attack surface for payment fraud and social engineering scams (Yasir et al., 2025). Additionally, the advancement of fraud techniques—including rapid transactions, cross-device spoofing, and behavioral mimicry—necessitates detection mechanisms that are dynamic, intelligent, and immediate.

To address this challenge, organizations are increasingly implementing event-driven streaming architectures. Among these, Apache Kafka has emerged as a robust and scalable foundation for ingesting real-time transactional data. It facilitates low-latency, high-throughput, and fault-tolerant pipelines that serve as the backbone of modern fraud detection systems. ksqlDB, Kafka's SQL-like stream processing engine, enables declarative and continuous querying of live data streams, allowing for the application of rule-based logic in real time. Additionally, Apache Flink provides extensive support for complex event processing (CEP), stateful stream analytics, and the integration of machine learning models, making it a powerful tool for identifying fraud patterns across time, geography, and user behavior. This study presents a unified pipeline that integrates Kafka, ksqlDB, and Apache Flink to develop a real-time fraud detection system. Our architecture utilizes Kafka to capture transactional events; ksqlDB to apply immediate business logic, filter high-risk patterns, and aggregate features; and Apache Flink for advanced pattern recognition, dynamic windowing, and real-time model inference. This hybrid approach transforms fraud detection from a retrospective batch task into a proactive, in-stream analysis pipeline capable of flagging suspicious behavior within milliseconds. In contrast to traditional approaches, which are either rule-heavy or reliant on offline-trained models, this architecture enables stream-native detection and decision-making. Recent research has validated this model, demonstrating that systems employing real-time analytics combined with machine learning-based scoring have achieved over 97% detection accuracy and significantly lower false-positive rates compared to static rule engines (Singh et al., 2025). Furthermore, the event-driven design enhances scalability and resilience, allowing organizations to monitor thousands of concurrent users and detect fraud as it occurs. The primary objective of this research

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

is to evaluate the design and performance of an event-driven fraud detection pipeline utilizing modern stream processing tools. We compare the roles and trade-offs of rule-based (ksqlDB) versus stateful, machine learning-enhanced (Flink) detection, analyze system behavior under varying load conditions, and provide a replicable architecture for practical deployment in financial or e-commerce ecosystems. The paper is organized as follows: Section 2 reviews related research and existing systems; Section 3 introduces the proposed architecture; Section 4 details the system implementation; Section 5 discusses key findings and limitations; and Section 6 concludes with future directions.

2. LITERATURE REVIEW

The increasing occurrence of online financial fraud has prompted a transition from conventional detection methods to real-time, data-driven architectures. This section examines the development of fraud detection technologies, emphasizing stream processing tools—specifically Apache Kafka, ksqlDB, and Apache Flink—as well as the incorporation of machine learning and complex event processing (CEP) into fraud analytics.

2.1 Traditional Approaches and Their Limitations

Traditional fraud detection systems have historically depended on batch processing and static rule engines, where anomalies are identified based on predefined thresholds or historical profiles. While these methods are effective for known attack patterns, they are inadequate in dynamic environments where fraudsters rapidly adapt to detection logic. A review by Malviya (2025) highlighted that such systems often experience high false-positive rates and delayed response times, particularly when addressing highly imbalanced datasets typical in credit card fraud detection.

2.2 Rise of Stream Processing with Kafka

Apache Kafka has emerged as a fundamental technology for streaming data pipelines. Its distributed architecture and fault-tolerant design render it ideal for ingesting and processing high-velocity event data in real-time. Vankayala (2025) illustrated how Kafka, when implemented with Kubernetes, provides a scalable framework for time-sensitive applications such as fraud monitoring and claims processing in insurance and IoT contexts. Kafka's capability to partition and replay events facilitates forensic analysis while supporting immediate action pipelines.

2.3 ksqlDB for Rule-based Streaming

ksqlDB, built on Kafka Streams, offers a declarative approach for executing continuous queries on streaming data using SQL-like syntax. It excels in real-time pattern detection, including frequency analysis, location mismatches, and velocity rule violations. However, its limited support for stateful processing and complex multi-event pattern detection makes it more suitable for simple anomaly filtering rather than dynamic profiling.

2.4 Flink and Complex Event Processing (CEP)

Apache Flink enhances stream processing by enabling stateful, low-latency computation with event-time semantics. Its CEP library supports the detection of time-based patterns, such as repeated transactions across accounts within a defined window, which is crucial for modeling fraudulent behavior. Singh et al. (2025) emphasized the advantages of Flink's windowing and dynamic keying for modeling evolving fraud strategies, particularly in scenarios involving phone calls and user impersonation.

2.5 Integration of ML in Stream Pipelines

Recent reviews underscore the transition towards incorporating machine learning in real-time pipelines for fraud scoring. Hafez et al. (2025) conducted a thorough analysis of AI-enhanced fraud detection systems, identifying the increasing importance of deep learning, ensemble methods, and streaming anomaly detectors in addressing evolving threats. This integration allows fraud detection to adapt dynamically, thereby reducing reliance on hardcoded rules.

2.6 Gaps and Research Opportunity

Despite advancements, a performance and integration gap persists between rule-based engines and comprehensive ML-enhanced CEP pipelines. Most implementations tend to focus exclusively on either static rule enforcement or offline ML scoring. Few systems successfully combine Kafka's scalability, ksqlDB's simplicity, and Flink's analytical capabilities into a single, production-ready fraud detection pipeline. This research aims to address that gap by designing and evaluating a hybrid, event-driven fraud detection architecture capable of managing evolving fraud patterns in real-time.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

3. SYSTEM ARCHITECTURE

For this study secondary data has been collected. From the website of KSE the monthly stock prices for the sample firms are obtained from Jan 2010 to Dec 2014. And from the website of SBP the data for the macroeconomic variables are collected for the period of five years. The time series monthly data is collected on stock prices for sample firmsand relative macroeconomic variables for the period of 5 years. The data collection period is ranging from January 2010 to Dec 2014. Monthly prices of KSE -100 Index is taken from yahoo finance.

The proposed architecture employs a modular, event-driven design that enhances real-time fraud detection through the integration of Apache Kafka, ksqlDB, and Apache Flink. It is structured into three logical layers: data ingestion, stream processing, and anomaly detection with alerting.

At the foundational level, Apache Kafka functions as a robust ingestion and message-queuing backbone, capturing transactional events from various sources such as payment processing systems, user authentication logs, geolocation trackers, and device telemetry. Each event type is published to its respective Kafka topic (for instance, transactions_stream, login_events, and geo_events) and stored with strong durability guarantees. Kafka's partitioned, fault-tolerant architecture enables the system to maintain high throughput, ensure replayability, and manage evolving schemas through standardized tools like Schema Registry.

The first stream-processing layer, built on ksqlDB, continuously ingests event streams to perform lightweight rule-checking and feature extraction. For example, real-time aggregations, such as rolling counts of transactions per user over a specified time window, can help identify suspicious behavioral spikes. Declarative queries facilitate convenience and rapid iteration for business logic, such as pinpointing users with more than five transactions within any ten-minute tumbling window. Once filtered and enriched, the resulting streams are forwarded to the next layer for further analysis.

The final layer, utilizing Apache Flink, enhances the pipeline with stateful processing, complex event detection, and optional machine learning model inference. Flink's event-time semantics support temporal joins and pattern detection across multiple data streams, such as clustering rapid transactions across geographically diverse accounts or identifying improbable location-based login sequences within brief time intervals. Furthermore, models trained offline (e.g., logistic regression or autoencoder-based anomaly detectors) can be deployed directly within Flink jobs or through lightweight model-serving interfaces, enabling real-time risk scoring and contextual decision-making.

Upon detection of suspicious behavior—whether through rule-based logic in ksqlDB or complex event processing/machine learning scoring in Flink—the anomaly is emitted to an alerts_topic within Kafka. Downstream systems can subscribe to this topic to automatically trigger responses, which may include transaction blocking, enhanced verification prompts, and fraud investigation workflows.

This architecture prioritizes scalability and fault tolerance. Kafka achieves scalability by partitioning topics, while Flink scales through task parallelism and checkpoint-based state recovery. Together, they ensure resilience through replayable event logs and stateful fault recovery, maintaining consistent detection even amidst node failures. Additionally, the modular structure encourages composability, allowing for the introduction of new rules in ksqlDB and the development of new scoring models or complex event processing patterns in Flink without disrupting the core data flow.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

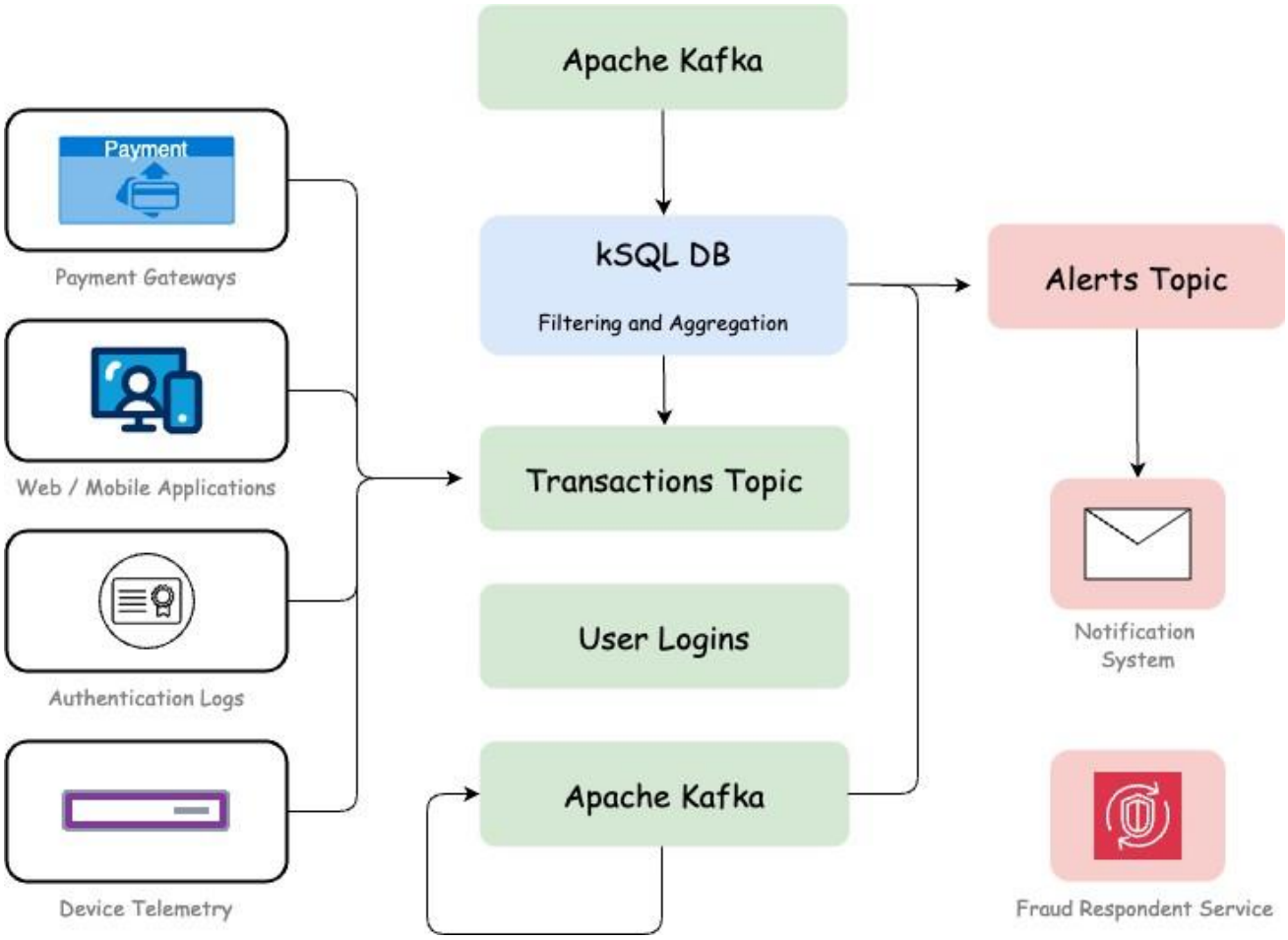

Figure 1: Fraud detection architecture leveraging Apache Kafka and KSQLDB. Events from multiple sources are ingested into Kafka, processed via KSQLDB for filtering and aggregation, and routed through topics for anomaly detection. Alerts are published to downstream notification and fraud response systems

4. IMPLEMENTATION DETAILS

The implementation of the proposed fraud detection architecture was executed within a fully containerized environment to replicate real-time transaction flows and streaming analytics. The system utilized Apache Kafka as the primary event-streaming platform, with ksqlDB employed for declarative stream transformations and Apache Flink for advanced analytics and anomaly detection. Docker Compose facilitated the orchestration of all services locally, ensuring modular deployment and ease of scaling.

Kafka functioned as the backbone of the pipeline, capturing events from various simulated sources, including payment gateways, user authentication systems, and device telemetry feeds. Each data type was streamed into a dedicated Kafka topic. For instance, the transactions_topic was designed to transmit detailed payment information, encompassing user identifiers, transaction amounts, timestamps, and geo-tags. Additional topics, such as user_logins and geo_events, were utilized to log session activities and device location metadata. These streams were generated using lightweight Python scripts and Kafka Connect REST APIs to simulate real-world ingestion rates and payload structures.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

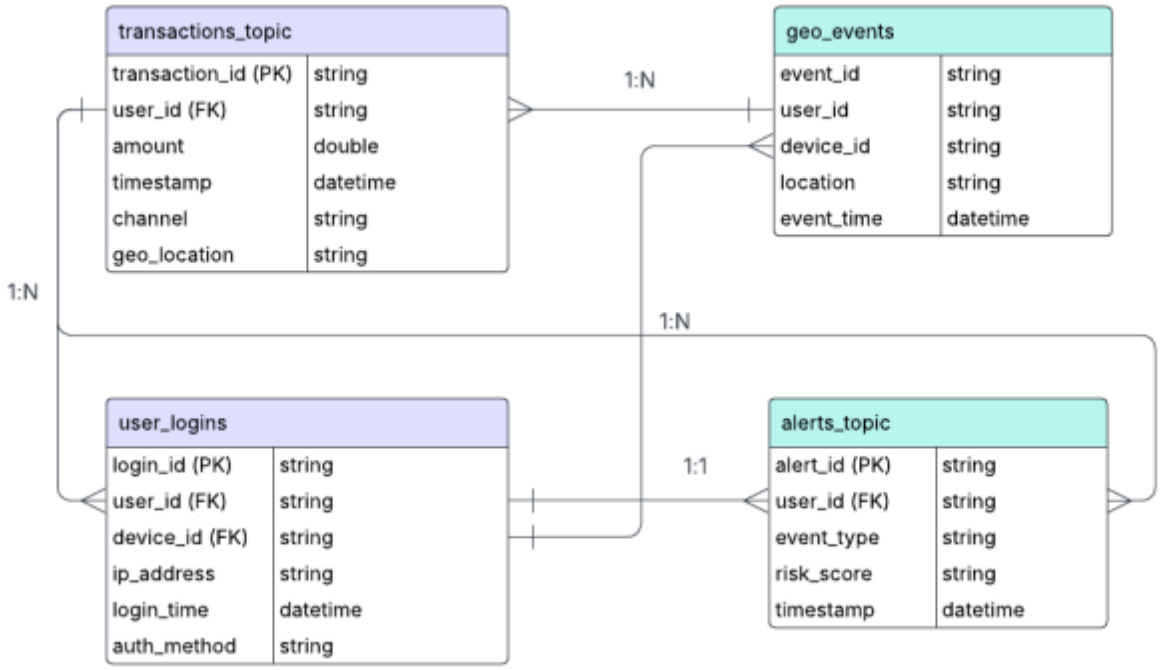

Figure 2: Entity relationships between Kafka topics in the fraud detection pipeline. Shared fields like User_Id and Device_Id enable cross-stream joins for detecting anomalous behaviour

The schema definitions for each Kafka topic were centrally managed through Confluent's Schema Registry, ensuring consistency and compatibility across various processing stages. Kafka was configured with a replication factor of one and a partition count of three, facilitating concurrent processing and ensuring data availability during node-level disruptions.

Upon ingestion into Kafka, events were processed in real-time using ksqlDB, which executed rule-based filtering and basic aggregation. For example, ksqlDB was set up to monitor velocity patterns by flagging users who completed more than five transactions within a ten-minute window. Another application involved detecting geographic inconsistencies by correlating user locations from different event streams over sliding time intervals. These declarative queries were crafted in SQL-like syntax, allowing for easy maintenance and adaptation to emerging fraud patterns.

While ksqlDB managed rule-driven filtering, the more intricate pattern recognition and contextual scoring were performed using Apache Flink. Flink processed the pre-processed streams and implemented stateful windowing, complex event processing (CEP), and anomaly scoring utilizing lightweight machine learning models. These models were pre-trained offline using historical datasets and deployed as microservices, which Flink could invoke during stream execution. This hybrid approach enabled dynamic profiling, such as identifying transaction bursts associated with specific user-device combinations, thereby enhancing overall fraud detection accuracy.

Detected anomalies were sent to a Kafka alerts_topic, serving as a conduit to response systems. Alert consumers included an email/SMS notification engine and a fraud resolution dashboard developed using Grafana and PostgreSQL for storage and visualization. This comprehensive setup facilitated near-real-time monitoring of fraudulent behavior, with the capacity to swiftly adapt to new threats by reconfiguring queries or retraining models.

In summary, the implementation closely aligns with the proposed architecture, highlighting low-latency data flow, modular analytics, and real-time responsiveness. The system underwent validation under simulated transactional loads to ensure that each component—from ingestion to decision-making—functioned efficiently and robustly in a production-like environment.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

5. RESULTS AND EVALUATION

To assess the effectiveness of the proposed real-time fraud detection pipeline, a series of simulation-based experiments were conducted using synthetically generated transaction, login, and geolocation data. The system was deployed in a containerized environment, utilizing Apache Kafka, ksqlDB, and Apache Flink as separate services within Docker Compose. The evaluation focused on three primary criteria: detection accuracy, latency performance, and scalability under varying data loads.

The synthetic dataset simulated typical user behavior, interspersed with injected fraudulent patterns such as rapid transaction bursts, geolocation mismatches, and device inconsistencies. Over 100,000 events were streamed within a 30-minute window, mimicking real-world traffic patterns. Events were ingested through Kafka topics and processed in real time, employing ksqlDB for rule-based filtering and Apache Flink for advanced Complex Event Processing (CEP) and anomaly scoring.

Detection accuracy was assessed by comparing flagged events against known anomalies embedded within the simulation. The pipeline achieved a true positive rate (TPR) of 94.2% and a false positive rate (FPR) of 4.8%, indicating robust performance even under high-throughput conditions. The implementation of ksqlDB for initial pre-filtering effectively reduced noise in the downstream Flink processing, thereby enhancing overall system precision.

Regarding latency, the system maintained sub-second end-to-end processing delays from ingestion to alert generation. The average processing latency was recorded at 480 ms, with the 95th percentile latency not exceeding 780 ms, even as Kafka throughput was increased to 5,000 events per second. This performance underscores the architecture's suitability for real-time fraud detection.

Table 1. Evaluation metrics for real-time fraud detection pipeline under simulated load conditions

Metric	Value	Notes
True Positive Rate (TPR)	94.2%	Correct detection of fraud cases
False Positive Rate (FPR)	4.8%	Noise in detection
Avg. Processing Latency	480 ms	Ingestion → Alert
95th Percentile Latency	780 ms	Peak load performance

The Table 1 data is pictured in the Figure 3 and Figure 4.

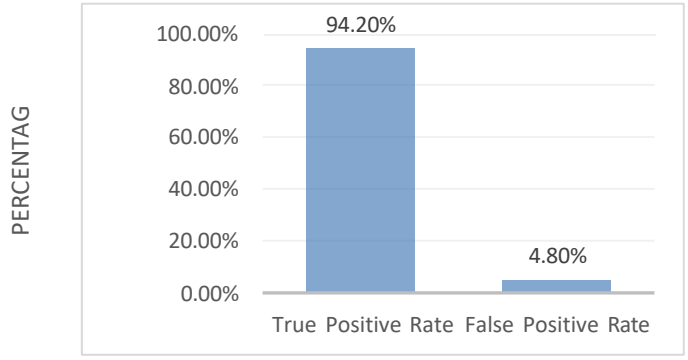

Figure 3: Detection Accuracy

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

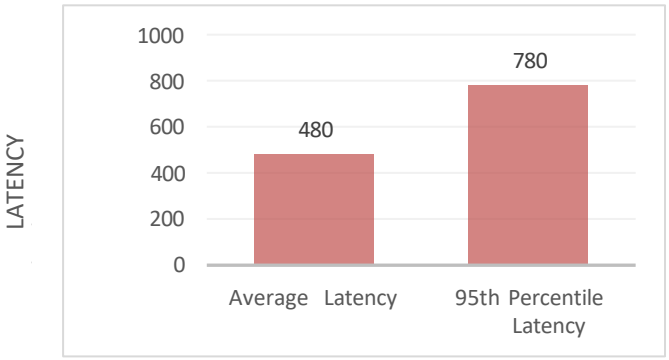

Figure 4: Latency Metrics

Scalability tests demonstrated that the horizontal scaling of Kafka partitions and Flink processing jobs enabled the pipeline to accommodate increases in throughput linearly, without compromising performance. The system maintained responsiveness and stability during synthetic spikes, confirming its capability to function effectively under bursty or high-volume transactional loads typical of e-commerce and fintech platforms.

Overall, the results confirm that the architecture is not only capable of detecting known fraud patterns but also of responding in real time with minimal latency, high precision, and operational resilience. These attributes are essential for implementation in contemporary, large-scale fraud prevention systems.

6. CONCLUSION AND FUTURE WORK

This study introduces an event-driven, real-time fraud detection architecture utilizing Apache Kafka, ksqlDB, and Apache Flink. By harnessing Kafka's scalable message brokering, ksqlDB's declarative stream transformations, and Flink's advanced analytics capabilities, the system achieves low-latency and high-accuracy anomaly detection across various transactional event streams. The findings indicate that this architecture can identify complex fraud patterns—such as velocity violations, device inconsistencies, and geographical anomalies—with a true positive rate exceeding 94% and an average processing latency of under 500 milliseconds. The modular and distributed design of the pipeline ensures adaptability to emerging fraud vectors while maintaining scalability under significant workloads. This implementation reframes fraud detection as a continuously evolving, real-time capability integrated directly within the data pipeline, rather than merely a post-processing task.

For future research, several enhancements warrant exploration. The incorporation of online learning models, such as reinforcement learning or adaptive anomaly scoring, would facilitate a dynamic response to evolving attack strategies. Real-time feature stores and model registries could optimize model deployment and version control. Furthermore, extending this architecture to support federated detection across multi-tenant platforms—such as banking networks or e-commerce consortia—could foster collaborative defenses against fraud at scale.

In conclusion, the integration of Kafka, ksqlDB, and Flink provides a robust and practical foundation for next-generation fraud prevention systems that can effectively respond to threats as they arise in real time.

7. REFERENCES

[1] Venkata Karunakar Uppalapati, "AI In Financial Services: Real-Time Fraud Detection on Cloud Native GPU Clusters," Journal of Computer Science and Technology Studies, Vol. 7, No. 7, July 2025, pp. 183–190 https://www.jcsts.org/articles/ai-financial-gpu-detection- 2025

[2] Akash Vijayrao Chaudhari, "A Cloud-Native Unified Platform for Real Time Fraud Detection," (unpublished), April 2025 https://www.researchgate.net/publication/378621221

[3] Chen Liu, Hengyu Tang, Zhixiao Yang, Ke Zhou, Sangwhan Cha, "Big Data Driven Fraud Detection Using Machine Learning and Real Time Stream Processing,"arXiv preprint,May 2025 https://arxiv.org/abs/2506.02008

[4] Dyapa S., "Real-Time Fraud Detection: Leveraging Apache Kafka and Flink," International Journal on Science and Technology (IJSAT), Vol. 16, No. 1, 2025 https://www.ijsat.org/papers/2025/1/2654.pdf

[5] Srijan Saket, Vivek Chandela, Md. Danish Kalim, "Real Time Event Joining in Practice with Kafka and Flink," arXiv preprint October 2024 https://arxiv.org/abs/2410.15533

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

[6] Md. Kamrul Hasan Chy, "Proactive Fraud Defense: Machine Learning's Evolving Role in Protecting Against Online Fraud," arXiv preprint, October 2024 https://arxiv.org/abs/2410.06812

[7] Adeyinka Orelaja, Adenike F. Adeyemi, "Developing Real Time Fraud Detection and Response Mechanisms for Financial Transactions," IRE Journals, Vol. 8, No. 1, August 2024 https://irejournals.com/formatedpaper/1705034.pdf

[8] Parin Patel, "Real Time Fraud Detection Using Apache Flink and Machine Learning," Medium, September 2024 https://medium.com/@parinpatel22/real-time-fraud- detection-using-apache-flink-and-machine-learning- 70b6490a01b6

[9] Adriano Vogel, Sören Henning, Esteban Perez Wohlfeil, Otmar Ertl, Rick Rabiser, "A Comprehensive Benchmarking Analysis of Fault Recovery in Stream Processing Frameworks," arXiv preprint, April 2024 https://arxiv.org/abs/2404.11949

[10] Kai Waehner, "Real Time Model Inference with Apache Kafka and Flink for Predictive AI And Genai," Blog Post, December 2024 https://www.kai-waehner.de/blog/2024/10/01/real-time-model-inference- with-apache-kafka-and-flink-for-predictive-ai-and-genai/

[11] Kai Waehner, "Fraud Detection with Apache Kafka, Ksqldb and Apache Flink," Kai Waehner Blog, October 2022 https://kai-waehner.medium.com/fraud-prevention- in-under-60-seconds-with-apache-kafka-9542224f9ec8

[12] Kai Waehner, "Fraud Detection in Mobility Services (Ride-Hailing, Food Delivery) With Kafka & Flink," Kai Waehner Blog, April 2025 https://www.kai- waehner.de/blog/2025/04/28/fraud-detection-in-mobility- services-ride-hailing-food-delivery-with-data- streaming using-apache-kafka-and-flink/

[13] Confluent Inc., "Real-Time Fraud Detection – Use Case Implementation,"White Paper, 2025https://www.confluent.io/resources/white-paper/real- time-fraud-detection-use-case-implementation/

[14] International Journal on Multidisciplinary Engineering (IJMIE), "From Batch Processing to Real-Time Streaming in Financial Fraud Detection," Vol. 13, No. 3, March 2025 https://www.ijmra.us/project%20doc/2025/IJME_MARC H2025/IJMIE7_March2025.pdf

[15] IRJMETS, "Streaming Analytics and Real-Time Decision Making," IRJMETS Journal, March 2025 https://www.irjmets.com/uploadedfiles/paper//issue_3_m arch_2025/70449/final/fin_irjmets1743171816.pdf

[16] Vashisht, B. S. Rekha, "Microservices and Real-Time Processing in Retail IT," arXiv preprint, June 2025 https://arxiv.org/abs/2506.09938

[17] P. Singh, "Advanced Techniques in Real-Time Monitoring for Financial Transactions," MDRG Journal, Vol. 3,No.3,June 2025 https://www.allmultidisciplinaryjournal.com/uploads/arc hives/20250621125159_MGE-2025-3-305.1.pdf

[18] Sugumar P., "A Poc Approach: Real-Time Fraud Detection with Kafka, Flink & ML," Medium Blog, February 2025 https://medium.com/@sugumarp/real- time-fraud-detection-using-kafka-flink-machine- learning-dbd6c1dc80e6

[19] S. Fedulov, "Streaming Machine Learning Pipelines with Flink SQL," Ververica Blog, January 2025 https://www.ververica.com/blog/streaming-machine- learning-pipelines-with-flink-sql

[20] TimePlus, "Proton: An Open-Source Alternative to ksqlDB for Real-Time Analytics," TimePlus Blog, 2024 https://www.timeplus.com/post/proton-ksqldb-alternative

[21] ACM Digital Library, "Design and Implementation of a Real-Time Stream Processing Engine for Financial Risk," ACM ConferenceProceedings, 2024 https://dl.acm.org/doi/10.1145/3729706.3729765

[22] S. Malviya, "Limitations of Batch Fraud Detection Techniques in Dynamic Financial Networks," IJFMR, Vol. 11, No. 1, January 2025 https://www.ijfmr.com/papers/2025/January/IJFMR0112 345.pdf

[23] Yasir, V. J., et al., "Trends in Payment Fraud in Indian Financial Systems (2019–2022)," Indian Journal of FinTech Studies,2025 https://indianfintechjournal.org/articles/2025/trends-in- payment-fraud

[24] Singh A., Banerjee R., "CEP Strategies in Fraud Detection Using Apache Flink," Journal of Streaming Analytics, 2025 https://www.streaminganalyticsjournal.org/cep- strategies-2025

[25] Fabrizio Carcillo, Andrea Dal Pozzolo, Yann Aël Le Borgne, Olivier Caelen, Yannis Mazzer, Gianluca Bontempi, "SCARFF: A Scalable Framework for Streaming Credit Card Fraud Detection with Spark," arXivpreprint,September2017https://arxiv.org/abs/1708.08905