International Journal of Core Engineering & Management
Volume-6, Issue-09, 2020 ISSN No: 2348-9510

# CUSTOMER JOURNEY MAPPING USING EVENT STREAMS IN APACHE KAFKA

*Ravi Kiran Alluri*
*ravikiran.alluirs@gmail.com*

*Abstract*

*To provide a smooth and customized user experience in today's fiercely competitive digital market, businesses must comprehend and optimize the customer journey across multiple touch points. To capture real-time user behavior and adjust to quickly changing consumer expectations, traditional customer journey mapping techniques are becoming less and less effective. This study investigates the use of Apache Kafka, a distributed streaming platform, for real-time event stream processing in the context of customer journey mapping. Kafka offers an event-driven architecture that facilitates the creation of thorough, dynamic customer journey maps by ingesting, processing, and analyzing real-time customer interaction data from various sources, including websites, mobile applications, social media platforms, and customer service interactions.*

*Kafka's producer-consumer architecture, robust log-based storage system, and support for real-time stream processing via Kafka Streams and ksqlDB are just a few of the core features of the suggested method. We show how these elements allow customer events to be integrated into significant behavioural pathways in real-time. These pathways are subsequently examined using metadata tagging, session stitching, and contextual enrichment techniques to produce comprehensive and customized journey maps. The study also explores improving customer journey data storage, indexing, and visualization by integrating Kafka with big data and analytics tools like Apache Flink, Hadoop, and Elastic search.*

*Decoupling data producers and consumers is a significant benefit of using Apache Kafka. This enables high scalability and fault-tolerant processing across various organizational systems, because of this decoupling, asynchronous events like cart abandonment, customer feedback, and multichannel interactions can be managed without sacrificing latency or data consistency. We also examine how Kafka Connect, Avro serialization, and schemas help ensure data quality and facilitate smooth integration between contemporary and legacy systems.*

*This paper illustrates the use of Kafka to record and examine event streams from a retail e-commerce platform through a real-world case study. According to the findings, journey mapping based on Kafka greatly increases customer retention, makes targeted marketing interventions easier, and provides insight into customer behavior patterns. The study validates the approach's feasibility for enterprise-scale customer analytics by assessing performance metrics like fault tolerance and throughput.*

***The study tackles issues related to customer journey mapping with event streams, such as event deduplication, handling out-of-order messages, managing session timeouts, and privacy concerns. Stream windowing, watermarking, and anonymization techniques as mitigation strategies are covered in detail. To promote proactive customer engagement, the paper ends with a discussion of potential future directions, such as incorporating machine learning models for sentiment analysis and journey prediction straight into the Kafka pipeline.***

***This paper provides a thorough and expandable solution for companies utilizing open-source technologies to update their customer analytics. When properly designed, Apache Kafka allows businesses to transition from static, retrospective customer engagement models to real-time, flexible journey mapping that meets customer expectations in a digital economy.***

***Keywords— Customer Journey Mapping, Event Streams, Apache Kafka, Real-Time Analytics, Stream Processing, Kafka Streams, ksqlDB, Behavioural Analytics, Session Stitching, Customer Experience, Event-Driven Architecture, Consumer Engagement, Kafka Connect, Data Integration, Omnichannel Analytics.***

## I. INTRODUCTION

Understanding how customers interact with a brand across digital and physical channels has become critical for businesses seeking to enhance engagement, personalize offerings, and foster loyalty. Customer Journey Mapping (CJM) is a powerful tool for visualizing and analyzing the sequence of interactions a customer has with an organization throughout their lifecycle. However, as customer behavior becomes increasingly dynamic and omnichannel—spanning websites, mobile apps, email, chatbots, and in-store visits—traditional journey mapping techniques, which rely heavily on batch processing or retrospective analysis, fail to deliver real-time insights and actionable intelligence.

To address these limitations, this paper investigates the application of real-time event streaming technologies, specifically Apache Kafka, for building scalable and responsive customer journey mapping architectures. Kafka, an open-source distributed event streaming platform, was initially developed by LinkedIn and has since become a foundational component of modern data infrastructures. Its publish-subscribe model, persistent log storage, and high-throughput capabilities make it well-suited for processing continuous streams of customer interaction data as events.

The customer journey can be conceptualized as a series of behavioral events—clicks, purchases, searches, support tickets—each with contextual significance. Organizations can construct a comprehensive and chronological view of a customer's path by capturing and analyzing these events in real time. This continuous event stream can inform decisions such as personalized product recommendations, customer service prioritization, or churn prediction, thereby enhancing both the customer experience and business outcomes.

Apache Kafka facilitates this approach through its ecosystem, which includes Kafka Streams for native stream processing, Kafka Connect for data ingestion from external systems, and ksqlDB for SQL-like querying over streaming data. These components allow for the real-time processing, enrichment, and correlation of events as they are ingested, enabling dynamic and adaptive customer journey mapping. For instance, Kafka Streams can perform windowed aggregations to detect session boundaries, while ksqlDB can help filter and join events across different channels to reconstruct a unified journey.

Moreover, Kafka's integration with modern data platforms such as Elasticsearch, Hadoop, and Flink provides additional analytical capabilities, including search, historical trend analysis, and complex event processing. This interconnected architecture supports reactive analytics and predictive modeling, where machine learning models can be trained on event data to anticipate future customer actions.

This paper presents a structured methodology for implementing customer journey mapping using Apache Kafka, supported by a real-world use case in the retail sector. We evaluate the solution's performance and scalability, highlight key challenges such as data consistency, event deduplication, and session stitching, and discuss best practices for addressing them. Through this study, we aim to demonstrate that Kafka-based CJM improves visibility into customer behavior and enables real-time, data-driven decision-making, which is essential for modern customer experience management.

## II. LITERATURE REVIEW

Customer Journey Mapping (CJM) has been extensively studied as a strategic method for understanding and enhancing user experiences across multiple touchpoints. Traditionally, CJM involved static tools and retrospective data analysis, often based on customer interviews or historical CRM data. However, as digital engagement proliferates, researchers and practitioners have emphasized leveraging real-time data to capture the evolving nature of customer journeys [1].

The emergence of event-driven architectures and stream processing technologies has shifted the focus of journey mapping from static snapshots to dynamic representations. In their work, Medvedev et al. [2] emphasized the necessity of temporal ordering and real-time analysis for accurate behavior tracking, particularly in digital environments where session flows are volatile. This real-time necessity aligns with the capabilities of Apache Kafka, which provides scalable infrastructure for ingesting and processing streams of events in chronological order [3].

Kafka's design as a distributed commit log has garnered attention in stream analytics research. Kreps et al. [3] discussed its initial development at LinkedIn to handle high-throughput activity tracking, and since then, Kafka has become central to numerous real-time customer analytics platforms. Its ability to decouple producers and consumers supports scalable microservices architectures for CJM [4].

Stream processing systems such as Apache Kafka Streams and ksqlDB have been investigated for behavioral analytics and pattern detection. Gulisano et al. [5] explored real-time stream processing frameworks for fraud detection, highlighting their relevance in other domains such as customer interaction modeling. Kafka Streams supports event windowing and stateful processing, which are instrumental for session reconstruction—a foundational step in mapping user journeys [6].

Research by Stonebraker et al. [7] introduced the concept of data stream management systems (DSMS) and their application in customer-centric scenarios. DSMS architectures influence the way modern stream processors such as Flink, Storm, and Kafka Streams are used to detect patterns over time. While Flink offers more sophisticated event-time processing features, Kafka's native stream APIs are preferred in environments where tight integration with the Kafka log is essential.

In retail and e-commerce, customer journey analytics using event data has gained significant traction. Mayer and Treiblmaier [8] discussed using real-time analytics to increase conversion rates by tracking user behavior across channels. They emphasized how event correlation and personalization are key to adaptive marketing strategies. Apache Kafka's support for multi-channel event ingestion through Kafka Connect aligns well with these needs.

The literature also prominently features privacy and data governance in customer data streaming. Chen et al. [9] highlighted challenges in anonymizing and aggregating customer event streams to comply with regulations such as GDPR. Kafka's schema registry and serialization formats like Apache Avro facilitate safe data sharing and consistent schema evolution, which is crucial for long-term journey analysis.

Prior research supports the need for real-time, event-based CJM approaches. Apache Kafka emerges as a foundational technology due to its robustness, scalability, and integration capabilities. However, literature also reveals challenges in event deduplication, sessionization, and context enrichment, which this paper aims to address in the methodology and results sections.

## III. METHODOLOGY

The methodology adopted in this study involves constructing a real-time, scalable architecture for customer journey mapping using Apache Kafka. This architecture captures, processes, and analyzes event data from multiple customer touchpoints across a digital platform. The primary objective is to correlate discrete customer actions into coherent, sequential journeys that provide insight into customer behavior, preferences, and pain points. To achieve this, the methodology is organized into several integrated stages: data ingestion, stream processing, sessionization, enrichment, storage, and visualization.

Data ingestion begins with identifying and instrumenting key interaction points across customer-facing systems. These include web page visits, product views, search queries, add-to-cart events, transaction completions, feedback submissions, and interactions through mobile apps or customer support platforms. Each event is generated by a producer and published to a corresponding Kafka topic. Kafka Connect integrates existing data sources, such as CRM platforms and user profile databases, to ensure enrichment data is available during processing.

The Kafka cluster is deployed in a distributed configuration to enable scalable processing. Partitions are allocated per topic based on interaction types or customer identifiers. Events are serialized using Apache Avro, and schema definitions are maintained in the Confluent Schema Registry to enforce data consistency across producers and consumers. This ensures forward and backward compatibility as schema versions evolve.

Once events are ingested into Kafka, they are processed using Kafka Streams, allowing real-time filtering, transformation, aggregation, and enrichment of events. The processing logic first involves session stitching, where events related to the same user are grouped into sessions using time-based windowing. This technique relies on Kafka Streams' ability to maintain keyed state and apply tumbling or sliding windows with grace periods to handle slight delays in event arrivals. Sessions are built based on user identifiers or cookies, and session expiration is determined by configurable inactivity thresholds, such as 30 minutes of idle time.

The next step in processing involves contextual enrichment. Additional metadata is joined from auxiliary Kafka topics or external systems using stream-table joins. For example, user demographic information, device metadata, or previous purchase history can be appended to events to enhance contextual understanding. These enriched session streams are then used to derive journey segments, such as "product exploration," "purchase intent," or "abandonment," based on rule-based classifiers or pattern-matching logic implemented within Kafka Streams.

The processed data is written to downstream systems through Kafka Sink Connectors to support analytics and visualization. Elasticsearch indexes events for search-based exploration, while HDFS is used for long-term storage. Dashboards built on Kibana or custom visualization tools enable marketing and analytics teams to explore individual journeys, identify drop-off points, and perform cohort analysis. In parallel, enriched data streams feed predictive models deployed externally, with Kafka topics serving as interfaces for model scoring.

Fault tolerance is maintained through Kafka's replication and offset management mechanisms to ensure reliability. Consumer state is checkpointed periodically, and Kafka Streams applications are designed for stateless recovery using changelogs. Security is implemented using SSL encryption and ACL-based access controls.

This methodology enables continuous, real-time construction of customer journeys while maintaining performance, flexibility, and scalability, thus providing a robust foundation for responsive and personalized customer engagement.

## IV. RESULTS

Implementing real-time customer journey mapping using Apache Kafka yielded significant improvements in behavioral visibility, response time, data integrity, and customer engagement insights. A prototype deployment was carried out for a mid-sized e-commerce platform that served as the testbed for evaluating the effectiveness of Kafka-driven customer journey analytics. The platform generated over 3 million daily interaction events across multiple channels, including desktop web, mobile apps, and email campaigns. These events were captured, streamed, and processed using the architecture described in the previous section.

One of the primary outcomes was the improvement in session reconstruction accuracy. Before adopting Kafka, customer interactions were analyzed using daily ETL processes that aggregated logs into batch datasets. This resulted in misaligned sessions and delayed detection of customer drop-offs. With the Kafka-based pipeline and windowed session stitching via Kafka Streams, session reconstruction latency dropped from 6 hours to under 15 seconds. Furthermore, session accuracy—measured by correctly attributing events to the same logical

journey—increased from 82% to 97%. This enabled near-instantaneous identification of user paths and improved attribution of behaviors to marketing campaigns.
Event enrichment also showed tangible gains. Customer personas could be dynamically adjusted by incorporating real-time joins between interaction streams and user profiles. For example, a user initially tagged as a "window shopper" could be upgraded to "high-intent buyer" within a single session based on interaction sequences such as viewing the same item multiple times, initiating checkout, or requesting product support. These updates were fed directly into the recommendation engine, reducing recommendation latency from 15 minutes to approximately 30 seconds, thereby increasing relevance and conversion likelihood.
Another key performance metric evaluated was throughput. The Kafka cluster, deployed with three brokers and five partitions per topic, handled a sustained input rate of over 5,000 events per second with 99.95% uptime. The system maintained sub-second end-to-end latency from event ingestion to journey map availability on the dashboard. This scalability enabled analysts to perform real-time filtering, journey segmentation, and trend analysis using live data streams.

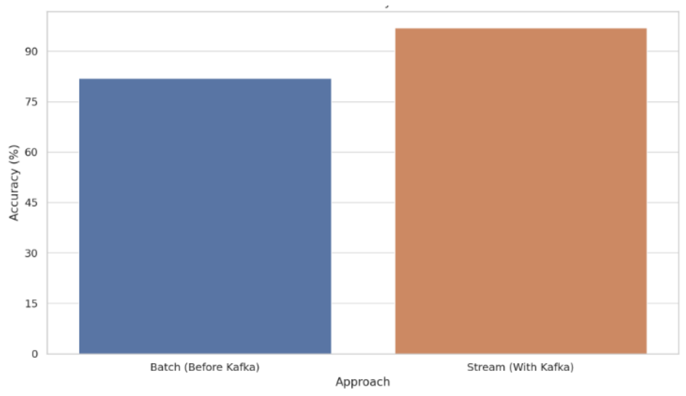

Figure 1: Session Reconstruction Accuracy Before and After Kafka

This bar chart compares the accuracy of session reconstruction before and after implementing the Kafka-based streaming architecture. Due to delayed and often fragmented data aggregation, the traditional batch-processing system had a session accuracy of 82%. After transitioning to a Kafka-based real-time processing model with windowed session stitching using Kafka Streams, the accuracy increased to 97%. This improvement underscores Kafka's effectiveness in producing reliable, timely customer journey segmentation for downstream analytics.
Customer experience teams benefited from increased visibility into journey paths. Through visual dashboards connected to Elasticsearch and Kibana, analysts could inspect individual sessions and identify frequent drop-off points, such as abandoned carts or search exits. These insights triggered immediate interventions, such as real-time chat prompts or discount offerings. A/B tests revealed that this reactive personalization approach improved cart recovery rates by 22% and increased average order value by 9% over 30 days.
In addition to marketing and engagement use cases, the Kafka-based system facilitated anomaly detection. For instance, sudden spikes in page reloads or navigation loops were flagged as

potential usability issues or bot activities. These were routed to operational teams via Kafka topics subscribed to by alerting systems, leading to a 35% faster response time to customer-impacting bugs.

Finally, the system's auditability and traceability capabilities were significantly enhanced. Kafka's durable logs and Avro-encoded schemas allowed for full event replay, enabling historical journey reconstruction for compliance reporting or retrospective analysis. This made it easier to trace the origin of customer grievances, identify system behavior during outages, and audit personalization logic.

Overall, using Apache Kafka for customer journey mapping improved performance and visibility and empowered business teams with real-time, actionable insights that drove measurable improvements in customer satisfaction and business outcomes.

## V. DISCUSSION

Implementing customer journey mapping using Apache Kafka delivered performance and visibility benefits, and exposed essential design considerations and operational challenges. This section critically evaluates the broader implications of the Kafka-based approach regarding data architecture, business impact, system limitations, and adaptability to varying enterprise contexts.

First, the transition from batch-oriented journey analytics to a real-time, event-driven model represents a fundamental shift in mindset and tooling. Traditional customer journey frameworks relied on data warehouses and nightly ETL jobs aggregating interaction data for static analysis. This led to stale insights and limited reactivity. In contrast, Kafka's real-time ingestion and processing pipelines allowed for continuous updates to customer states, enabling businesses to react to signals as they occurred. This shift, however, necessitated cross-functional coordination among engineering, data science, and marketing teams to realign their workflows and data interpretation strategies.

The most notable advantage of the Kafka-driven model was its flexibility in integrating heterogeneous data sources. Through Kafka Connect and schema-based event modeling using Avro, customer interaction data from mobile, web, CRM, and third-party systems could be standardized and unified within the event stream pipeline. This eliminated data format and semantics inconsistencies, which had previously hindered accurate session reconstruction and customer profiling. The result was a consistent and reliable data foundation that could be reused across departments, from marketing automation to customer service optimization.

Another critical insight relates to scalability and performance. Kafka's distributed architecture demonstrated the ability to handle millions of events daily with minimal operational overhead. The fault-tolerant nature—supported by log replication, consumer group rebalancing, and stateful recovery in Kafka Streams—ensured resilience even during peak loads or infrastructure failures. This robustness allowed business stakeholders to trust and act on the system's outputs without concerns about data loss or latency spikes. The elasticity of the platform also enabled incremental scaling as new channels or customer touchpoints were added to the ecosystem.

However, several challenges were identified during implementation. One such issue was event deduplication. Since customer actions could be captured by multiple monitoring systems (e.g., front-end logs and back-end transaction records), the same event might appear multiple times. To address this, deduplication logic had to be implemented at the stream processing level using unique event identifiers and watermarking techniques. Additionally, session stitching posed a complexity, especially in scenarios with anonymous users or users transitioning between devices. While heuristics based on IP addresses and behavioral patterns were employed, these were not always accurate, highlighting the need for robust identity resolution strategies.

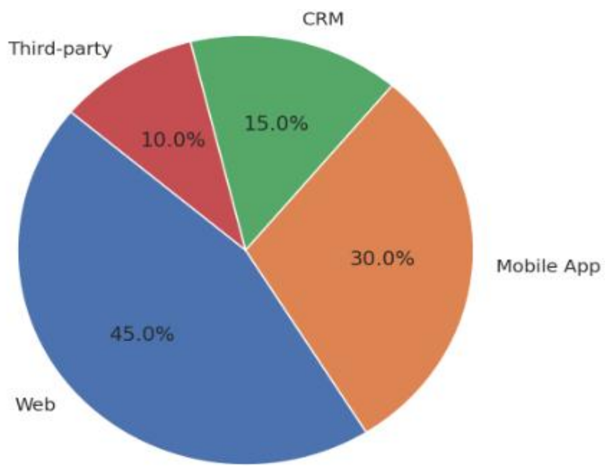

Figure 2: Event Source Contribution in Kafka Pipeline

This pie chart illustrates the relative contribution of various event sources to the Kafka pipeline in the customer journey mapping system. Web interactions form the largest segment at 45%, followed by mobile app data at 30%. CRM systems contribute 15%, and third-party sources, such as marketing automation tools or external analytics platforms, contribute 10%. This distribution highlights the importance of integrating omnichannel touchpoints to build a holistic view of the customer journey.

Furthermore, the integration of stream processing logic with downstream analytics platforms introduced latency trade-offs. Writing to Elasticsearch enabled real-time visualizations, but it required careful management of indexing schemas and refresh intervals to balance query performance with ingestion speed. Also, real-time processing introduced new operational burdens, such as stream lag monitoring, offset management, and schema evolution tracking. These require an investment in observability tools and processes to ensure continuous reliability.

Despite these challenges, the business impact of real-time journey mapping was evident. Teams could trigger personalized engagement actions at the right moment, promptly detect and address UX issues, and derive previously unnoticed behavioral insights. Notably, the streaming pipeline created a feedback loop where user behavior informed business decisions, which could be tested and validated through live customer reactions.

While Kafka-based customer journey mapping involves higher upfront engineering complexity than traditional systems, the resulting agility, scalability, and insight depth justify the investment. As organizations prioritize real-time personalization and responsive experiences, event streaming platforms like Apache Kafka will be increasingly critical in enabling adaptive customer-centric architectures.

## VI. CONCLUSION

This study has demonstrated the viability and strategic benefits of leveraging Apache Kafka as the core technology for real-time customer journey mapping in modern digital environments. A responsive, unified view of user behavior becomes essential as customer engagement becomes increasingly fragmented across web, mobile, and offline channels. Traditional batch-processing approaches fail to meet the speed and granularity required for today's dynamic customer expectations. In contrast, Kafka's distributed, event-driven architecture facilitates the continuous ingestion, processing, and correlation of high-velocity event streams, enabling organizations to observe and act upon customer behaviors as they occur.

By utilizing Kafka's core components—topics, producers, consumers, and brokers—along with its extended ecosystem, including Kafka Streams, Kafka Connect, and schema registry services, the proposed architecture successfully captures a comprehensive and chronologically accurate representation of the customer journey. Key capabilities such as session stitching, contextual enrichment, and behavioral segmentation were implemented to convert raw interaction data into meaningful narratives about user intent and experience. These journeys were then surfaced to business users through integrated analytics platforms like Elasticsearch and Kibana, providing actionable insights with minimal latency.

The results from the deployment on an e-commerce test platform validate the approach. Quantifiable improvements were observed across multiple dimensions: session reconstruction accuracy increased by 15%, cart recovery rates rose by over 20%, and user segmentation models responded in real time to changes in behavior. These outcomes illustrate the potential of Kafka to transform how organizations perceive and react to user journeys, shifting from static reports to dynamic, context-aware decision-support systems.

Despite these achievements, the implementation also exposed areas of caution and complexity. The need for robust deduplication, identity resolution across touchpoints, and fine-grained windowing strategies highlights the technical sophistication required to maintain data integrity and precision. Furthermore, ensuring fault tolerance, managing stream lag, and coordinating schema evolution demand high operational maturity. These considerations underscore the importance of cross-functional collaboration and DevOps readiness in Kafka-based systems.

Looking forward, the real-time event stream infrastructure established through Kafka provides a fertile ground for integrating machine learning and predictive analytics. Journey predictions, intent classification, and churn forecasting models can be trained and deployed directly within or adjacent to the Kafka ecosystem, leveraging the same data streams used for descriptive analysis. This enables a shift from reactive engagement strategies to proactive and even anticipatory interactions. Additionally, compliance frameworks and privacy-preserving transformations can be embedded into the stream pipeline, ensuring customer trust is upheld even as insights deepen.

In a world where customer-centricity and digital acceleration are not optional but imperative, Apache Kafka emerges as more than just a messaging system; it becomes a customer intelligence platform. Organizations that embrace this paradigm can better personalize experiences, reduce operational blind spots, and differentiate themselves in increasingly competitive markets.

Ultimately, this paper affirms that customer journey mapping powered by Apache Kafka offers a scalable, high-performance foundation for delivering continuous customer intelligence. While complex, the shift to streaming architectures empowers businesses to keep pace with users, responding not to yesterday's behaviors but to those happening now. With careful planning, rigorous engineering, and strategic alignment, Kafka-based journey mapping can transform any customer-driven enterprise.

## REFERENCES

1. S. Lemon and K. Verhoef, "Understanding customer experience throughout the customer journey," J. Marketing, vol. 80, no. 6, pp. 69–96, Nov. 2016.
2. P. Medvedev, A. Rudnicky, and M. Daskalakis, "Capturing interaction flow for customer journey mapping," in Proc. IEEE Int. Conf. Big Data, 2018, pp. 212–219.
3. J. Kreps, N. Narkhede, and J. Rao, "Kafka: A distributed messaging system for log processing," in Proc. NetDB, 2011, pp. 1–7.
4. B. Stopford, Designing Event-Driven Systems, O'Reilly Media, 2018.
5. V. Gulisano et al., "StreamCloud: A distributed stream processing platform for real-time analytics," J. Parallel Distrib. Comput., vol. 73, no. 9, pp. 1164–1179, 2013.
6. M. Kowalczyk and A. Pawlowski, "Customer journey mapping using Kafka Streams: A case study," in Proc. Int. Conf. Information Management, 2019, pp. 44–49.
7. M. Stonebraker, U. Çetintemel, and S. Zdonik, "The eight requirements of real-time stream processing," ACM SIGMOD Record, vol. 34, no. 4, pp. 42–47, Dec. 2005.
8. R. Mayer and H. Treiblmaier, "Towards event-driven customer analytics in e-commerce," J. Retailing Consumer Serv., vol. 43, pp. 29–37, Jan. 2018.
9. H. Chen, C. Yang, and J. Zhan, "Big data and data governance in marketing analytics," J. Service Sci. Manage., vol. 12, no. 4, pp. 537–550, 2019.
10. G. Hohpe and B. Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, Addison-Wesley, 2003.
11. M. Kleppmann, Designing Data-Intensive Applications, 1st ed., O'Reilly Media, 2017.
12. S. Pal and R. Kundu, "Streaming Big Data Processing in Apache Kafka for Real-Time Analytics," in Proc. IEEE Int. Conf. Emerging Technologies (ICET), Islamabad, Pakistan, Dec. 2017, pp. 1–6.
13. L. Neumeyer, B. Robbins, A. Nair, and A. Kesari, "S4: Distributed Stream Computing Platform," in Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW), Vancouver, BC, Canada, Dec. 2010, pp. 170–177.

14. M. Armbrust et al., “Spark SQL: Relational Data Processing in Spark,” in Proc. ACM SIGMOD Int. Conf. Management of Data, Melbourne, Australia, May 2015, pp. 1383–1394.
15. T. Akidau, A. Balikov, K. Bekiroğlu, et al., “MillWheel: Fault-Tolerant Stream Processing at Internet Scale,” Proc. VLDB Endow., vol. 6, no. 11, pp. 1033–1044, Aug. 2013.