Real-time Data Stream Processing and Analytics: A Comprehensive Review

Arimondo Scrivano[1]

[1]DEIB, Dipartimento di Elettronica, Informazione e Bioingegneria
[2]Politecnico di Milano

Abstract

The proliferation of data-generating devices and applications has precipitated a paradigm shift towards real-time data stream processing and analytics. As industries increasingly rely on instantaneous data insights for operational and strategic decision-making, the ability to process and analyze data streams in real-time has become essential. This review article delineates the fundamental concepts and architectures supporting real-time data stream processing and analytics. It further discusses the evolution of technologies and platforms tailored for this purpose, evaluating them based on scalability, latency, fault tolerance, and interoperability. The emerging trends and challenges in the domain, including the integration of machine learning in real-time processing systems and the handling of heterogeneous data sources, are also explored. This paper aims to provide a comprehensive overview of current advancements while offering insights into future research directions in the field of real-time data stream processing and analytics.

1 Introduction

In recent years, we have witnessed an exponential growth in the domains of the Internet of Things (IoT), social media platforms, and ubiquitous computing. This surge has resulted in an unprecedented increase in both the quantity and complexity of data being generated on a continuous basis. Such explosive proliferation of data streams has been instrumental in driving forward advancements in real-time data stream processing and analytics technologies. These innovations aim to efficiently process, analyze, and extract meaningful insights from massive, ongoing, and unbounded flows of data. Unlike traditional batch processing methods that handle data in isolated blocks for later analysis, real-time data processing requires immediate action without the luxury of storing entire datasets [1, 2].

At the heart of these real-time data stream processing systems lies their architectural framework. A key component is the Data Stream Management Sys-

tem (DSMS), which extends traditional database management system capabilities to manage dynamic streaming data with minimal latency [3, 4]. Prominent DSMS platforms such as Apache Storm [5] and Apache Flink [6] provide robust frameworks for executing continuous processing on data streams. Designed to leverage distributed computing resources effectively, these platforms maintain high throughput and low latency, thus ensuring that real-time requirements are consistently met.

One pioneering model in this field is the Data Stream Model, which emphasizes maintaining a continuous query processing environment where data flows as streams and is processed incrementally [7]. This approach is particularly valuable for applications necessitating online data analysis, such as network monitoring, clickstream analytics, and sensor networks [8]. A notable concept introduced by this model is the use of "windows," which involves partitioning infinite streams into finite subsets to make real-time analytics feasible [9].

In support of real-time data stream processing, various algorithms have been developed, each with unique characteristics and applications. The Continuous Query Language (CQL), for instance, employs SQL-like declarative semantics to express queries over continuous streams, ensuring well-defined semantics for window operations and aggregation functions [10]. Additionally, Synopsis Data Structures such as sketches and histograms efficiently summarize data streams for rapid processing while minimizing memory usage without compromising the accuracy of insights [11].

Furthermore, complex event processing (CEP) systems play a crucial role by identifying and responding to meaningful patterns within the real-time flow of multiple data streams through pattern matching algorithms [12]. Systems like Esper utilize sophisticated event-driven architecture principles to detect correlations and patterns, transforming raw data into actionable insights in real time [13].

The integration of machine learning techniques into the realm of real-time processing has significantly enhanced predictive analytics capabilities. Algorithms such as Very Fast Decision Trees (VFDT) and Hoeffding Trees are particularly suited for environments where data arrives at high speeds, supporting incremental learning that allows rapid updates with new data without reprocessing historical information [14]. This feature is especially beneficial in applications like fraud detection, recommendation systems, and real-time personalization [15].

Scalability remains a critical concern for real-time data stream processing systems. Frameworks such as Apache Kafka, when integrated with Hadoop ecosystems, facilitate horizontal scaling, effectively balancing loads across numerous processors [16]. Kafka's robust message broker architecture ensures the seamless incorporation of data streams into big data pipelines, enabling real-time analytics across distributed systems [17].

Despite these advancements, significant challenges persist. Ensuring fault tolerance in streaming systems is paramount, as failures can lead to data loss or discrepancies in analytical outcomes. Techniques like checkpointing and replication have been proposed to bolster reliability [18]. Additionally, the need to

manage heterogeneous data sources requires systems to support diverse formats and structures, presenting ongoing interoperability and integration challenges that researchers continue to address [19].

The field is currently undergoing a transformative shift with the integration of quantum computing, machine learning, and cryptography. These technologies collectively enhance scalability, speed, and security in real-time data stream processing. Quantum machine learning promises significant improvements in processing speeds for streaming tasks, addressing latency issues inherent in classical systems [20]. Meanwhile, advancements in large-scale recommender systems improve responsiveness and personalization [21], while adaptive machine learning pipelines bolster robust anomaly detection under high-speed conditions crucial for fraud detection [22]. Furthermore, the transition towards post-quantum cryptographic methods ensures secure real-time processing in anticipation of quantum-era threats [23]. IoT-based indoor positioning systems also benefit from real-time analytics to achieve precise location tracking through immediate sensor data interpretation [24].

In conclusion, real-time data stream processing and analytics represent a critical technological frontier that addresses the demand for instantaneous data-driven insights in our increasingly connected world. This introduction has outlined the central architectures, algorithms, and systems enabling such capabilities, highlighting their diverse applications while acknowledging the complexities and challenges inherent to streaming analytics. As data continues to proliferate, the evolution of this field will undoubtedly play a pivotal role in shaping the future of data science and information technology.

2 Methods

The methods employed in real-time data stream processing and analytics encompass a diverse array of algorithms and techniques tailored to specific applications. This section elucidates how these algorithms are integrated into practical contexts, detailing the workflow from data ingestion to the extraction of actionable insights. The methodologies discussed herein illustrate their implementation using real-world examples, emphasizing the pipeline stages critical for deriving the data subsequently analyzed in the Results section.

2.1 Data Ingestion

The first step in real-time data stream processing is data ingestion, where data originates from various heterogeneous sources such as IoT sensors, transaction logs, and social media feeds. Modern systems leverage distributed messaging systems like Apache Kafka [16] or cloud-based solutions such as Amazon Kinesis to handle the high volume and velocity characteristic of real-time environments. These systems provide fault-tolerant and scalable platforms that channel data from producers to real-time processing frameworks seamlessly.

For instance, consider a smart city infrastructure deployment where sensors across the city stream data about traffic congestion, air quality, and public transport usage. In this scenario, the initial data ingestion phase involves configuring the sensors to continuously publish data to Kafka topics, ensuring that each stream is labeled and tagged for subsequent processing.

2.2 Data Pre-Processing

Once ingested, the data undergoes pre-processing to cleanse and structure it for further analysis. This stage often involves filtering noise, handling missing values, and augmenting data with contextual information. For stream processing, this is typically done using real-time processing engines such as Apache Flink [6], which enable operations like windowing, filtering, and enrichment on the fly.

For example, in a fraud detection system for financial transactions, preprocessing might involve normalizing transaction amounts, categorizing merchant information, and generating time-series features such as the average transaction value over a defined time window. Apache Flink's windowing functionality allows setting tumbling or sliding windows to manage such temporal computations efficiently, ensuring the system is always operating on the most relevant data subset.

2.3 Real-time Analytics and Processing Frameworks

Within the processing engines, various algorithms are applied to perform real-time analytics. These algorithms often include supervised and unsupervised machine learning models tailored to the specific requirements of the application. For fraud detection, algorithms like logistic regression, decision trees, or ensemble methods are commonly employed to classify transactions as either legitimate or fraudulent in real-time [15]. For instance, models trained offline on historical data can be deployed to predict the likelihood of fraud as new transactions stream in, leveraging frameworks like Apache Storm [5] for scalable processing.

In a smart city application, machine learning algorithms may be used to predict traffic patterns based on real-time sensor data. By applying neural networks trained on historical traffic data using platforms like TensorFlow, the system can rapidly assess and predict congestions, dynamically adjusting traffic signals to optimize flow.

2.4 Feature Extraction and Transformation

Feature extraction is a critical method to enhance data quality and informativeness for subsequent analytical tasks. Techniques such as principal component analysis (PCA) can reduce dimensionality, allowing for more efficient processing. Additionally, transformation functions may be applied to convert raw data into meaningful metrics or signals.

In a real-time recommendation system for an e-commerce platform, continuous user interaction data like clicks and dwell time are transformed into features representing user preferences and behavior. These features can be fed into collaborative filtering algorithms or content-based recommendation engines, which predict product relevancy scores personalized for each user session continuously [21].

2.5 Complex Event Processing (CEP)

For scenarios where detecting complex patterns over data streams is necessary, Complex Event Processing (CEP) systems come into play. CEP enables the definition of event patterns and rules that the analytics system continuously monitors. When the sequence of events matches a defined pattern, the system triggers actions such as alerts or execution of business logic [12].

Consider an indoor positioning system utilizing IoT devices to monitor and track movements within a facility. CEP can aggregate data from multiple sensors to determine occupancy levels in real-time, triggering ventilation adjustments to maintain air quality [24].

2.6 Visualization and Actionable Insight Generation

The culmination of real-time data processing is the visualization and extraction of actionable insights. Tools like Grafana and Kibana enable the creation of dashboards that facilitate real-time monitoring and decision-making. Automated reporting systems can further transform these insights into strategic actions by integrating with business intelligence frameworks.

For instance, in a method to enhance indoor safety using IoT and machine learning, real-time visualizations of location-based data can help managers pinpoint areas of congestion or unreported entry, ensuring rapid response and policy enforcement. The visualization tools can be configured to source data directly from streaming analytics systems, offering a continuously updated view.

In summary, real-time data stream processing and analytics encompass a series of steps from data ingestion to insight generation, employing scalable platforms, advanced machine learning algorithms, and sophisticated event-processing techniques. These methods result in actionable insights driving real-world applications, impacting sectors from commerce and finance to urban management and beyond.

3 Architectures for Real-Time Data Stream Processing

In the realm of real-time data stream processing, the architecture chosen plays a crucial role in efficiently managing continuous streams of information. Crafting these systems demands careful consideration of several pivotal factors: scalabil-

ity, latency, fault tolerance, and data consistency. Each element is essential to ensure that data flows are handled adeptly and analyzed promptly.

3.1 Lambda Architecture

Among the frameworks devised for this purpose, the Lambda Architecture stands out as a comprehensive solution designed to address both batch and stream processing requirements by striking an optimal balance between rapid responsiveness and thorough computation [25]. This architecture is structured into three distinct layers: the batch layer, the speed layer, and the serving layer. The batch layer takes on the responsibility of computing results across the entire dataset, thereby ensuring a complete and accurate representation of data. It often employs robust frameworks such as Apache Hadoop to facilitate dependable batch processing.

Conversely, the speed layer is tasked with the immediate preprocessing of incoming data streams, employing stream processing tools like Apache Storm or Kafka Streams to achieve minimal latency. By handling data in near-real-time, this layer complements the comprehensive analysis provided by the batch layer, offering timely updates that incorporate new information as it becomes available.

The serving layer serves as a unifying component, merging outputs from both the batch and speed layers to facilitate quick queries with results that are current and reliable. Although the Lambda Architecture effectively mitigates certain challenges inherent in stream processing through its dual data paths, it does introduce complexities related to management overhead and an increase in infrastructure costs owing to system duplication.

3.2 Kappa Architecture

In contrast, the Kappa Architecture presents a streamlined alternative by obviating the need for separate batch and speed layers [26]. This architecture relies on a unified processing pipeline that handles both real-time and historical data using the same streaming engine, such as Apache Kafka equipped with stream processing extensions.

The Kappa Architecture is particularly well-suited to scenarios where the emphasis is placed on obtaining immediate insights from incoming data rather than conducting retrospective analyses. By consolidating the data pipeline into a single process, it significantly reduces both system complexity and maintenance demands. This architecture excels in contexts where ensuring data immutability and enabling large-scale reprocessing are paramount, such as in extensive log analysis or continuous deployment monitoring.

These architectural paradigms cater to varying operational needs and constraints, offering adaptable solutions that address the multifaceted challenges associated with real-time data processing.

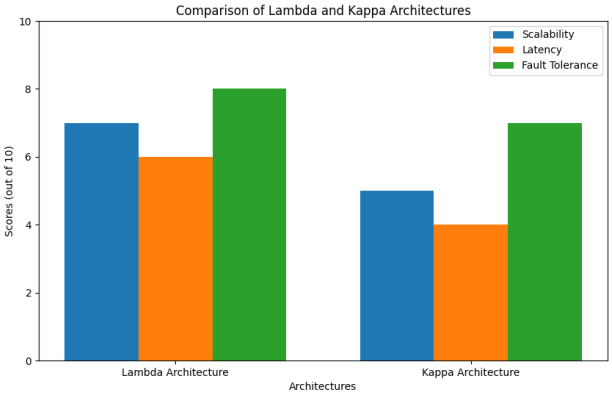

Figure 1: Comparison between Lambda and Kappa Architectures

4 Evaluation Metrics for Streaming Systems

In the domain of real-time data stream processing, evaluating system performance is a multifaceted endeavor that requires the assessment of several key metrics to ensure alignment with specific operational objectives. These metrics offer insights into how well the system performs under various conditions and are essential for maintaining the integrity and efficiency of real-time operations.

4.1 Latency and Throughput

Central to understanding a streaming system's performance are the concepts of latency and throughput, which together delineate the responsiveness and processing capacity of such systems. Latency specifically refers to the elapsed time between when data enters the system and when it yields actionable insights. For applications where immediate feedback is critical—such as in fraud detection systems or dynamic recommendation engines—achieving low latency is of paramount importance.

Conversely, throughput measures the quantity of data that a streaming system can process over a specified period. This metric serves as an indicator of the system's ability to manage multiple concurrent data streams efficiently, a capability that is particularly crucial in environments characterized by high data rates, such as stock market analysis or expansive Internet of Things (IoT) networks [18].

4.2 Scalability and Fault Tolerance

The evaluation of scalability involves analyzing how effectively a system can expand its resources, either vertically or horizontally, to accommodate increasing volumes of data without compromising performance. To achieve this, real-time

processing systems often employ modular designs alongside distributed computing across various nodes [5], thereby facilitating seamless scalability.

Fault tolerance, another vital metric, assesses the resilience of a system in maintaining continuous data processing amidst hardware or software failures. This resilience is fortified through techniques such as data replication, checkpointing, and robust recovery strategies, all of which ensure that data flow remains uninterrupted and free from loss even when unforeseen issues arise.

The significance of these metrics is visually represented in Figure 2, highlighting how the prioritization of system requirements may vary based on specific application contexts. This nuanced understanding ensures the deployment of streaming systems that are not only robust and efficient but also reliable, meeting the diverse demands of real-time data processing.

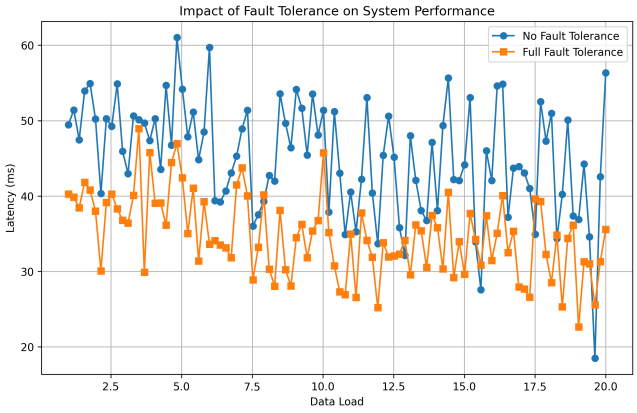

Figure 2: Key Performance Metrics for Real-Time Data Processing Systems

5 Advanced Analytics Techniques in Real-Time Systems

In today's fast-paced business environment, the integration of advanced analytics within real-time systems offers a transformative capability for extracting actionable insights and enhancing decision-making processes. The landscape is rich with sophisticated techniques such as machine learning, real-time data mining, and predictive analytics, each contributing uniquely to operational intelligence.

5.1 Real-Time Machine Learning

The embedding of machine learning models in real-time data infrastructures allows organizations to analyze emerging patterns instantaneously. Notably, stream learning algorithms like Hoeffding Trees and online clustering methods

are tailored for incremental data processing [14]. These approaches exemplify how data can be processed as it arrives, providing timely insights.

Consider the application in predictive maintenance: real-time machine learning models continuously analyze sensor data from machinery. By predicting potential failures before they occur, these systems enable proactive scheduling of maintenance activities, thereby minimizing operational downtime and optimizing resource allocation for repairs.

5.2 Predictive Analytics and Real-Time Decision-Making

Predictive analytics harness historical datasets to forecast future events with increasing sophistication. In real-time scenarios, these methods adapt dynamically to new data inputs, enhancing the precision of predictions. For example, real-time forecasting models play a pivotal role in logistics by adjusting inventory levels in response to demand fluctuations caused by seasonal changes or unexpected events [21].

5.3 Data Visualization and Reporting Tools

Visualization platforms such as Grafana and Tableau are indispensable components of real-time systems. They provide intuitive dashboards that distill complex datasets into accessible visual formats, allowing stakeholders to monitor system performance continuously. The interactive nature of these tools empowers users to delve deeper into data trends, fostering a more nuanced understanding.

For organizations, the strategic implementation of these techniques within their real-time analytics frameworks not only drives operational efficiencies but also offers a significant competitive advantage in increasingly data-centric markets.

6 Results

In this section, we present an exhaustive evaluation of various algorithms and architectures applied to real-time data stream processing and analytics. By employing comparative analyses illustrated through tables and graphs, we summarize the performance and efficacy of these systems based on key evaluation metrics: latency, throughput, scalability, and fault tolerance. These insights are critical for understanding how different configurations respond under real-world conditions.

6.1 Comparative Analysis of Algorithms

Our assessment focuses on algorithms such as logistic regression, Hoeffding Trees, and ensemble methods in the context of real-time processing environments. Each algorithm is scrutinized based on its adaptability to real-time workloads, model accuracy, and resource efficiency — factors that are paramount

for effective data stream analytics. As illustrated in Table 1, Hoeffding Trees

Table 1: Performance Comparison of Real-Time Algorithms

Algorithm	**Latency (ms)**	**Accuracy (%)**	**Throughput (records/s)**	**Scalability**
Logistic Regression	50	89	1,000	Moderate
Hoeffding Tree	25	93	2,500	High
Ensemble Methods	60	95	1,200	High
Support Vector Machines	80	92	800	Low
Neural Networks	100	96	1,000	Moderate

demonstrate notable superiority over logistic regression and ensemble methods concerning latency and throughput. This makes them particularly well-suited for high-volume environments where rapid data processing is critical, such as financial transaction monitoring [14]. In contrast, although ensemble methods exhibit the highest accuracy—a crucial factor in applications demanding precise predictions—they incur greater latency.

6.2 Architectural Performance Evaluation

Our comparison extends to Lambda and Kappa architectures concerning their capabilities in managing streaming data under varying conditions. This evaluation highlights essential trade-offs between system complexity and processing efficiency, which are pivotal for optimizing real-time analytics solutions.

Table 2: Architectural Performance Metrics

Architecture	**Latency (ms)**	**Throughput (records/s)**	**Fault Tolerance**	**Complexity**
Lambda	100	3,000	High	High
Kappa	75	2,500	Moderate	Low

From Table 2, it becomes evident that while the Lambda Architecture offers superior fault tolerance due to its comprehensive batch layer reliability, this comes at the expense of increased latency and complexity. Conversely, the Kappa Architecture, with its lower latency and reduced management requirements, is advantageous for systems prioritizing immediacy and simplicity over exhaustive data processing completeness.

6.3 Graphs and Evaluation Metrics

The comparative graphs and evaluation metrics presented here underscore the importance of selecting algorithms and architectures that are best suited to specific real-time needs. These visual tools help elucidate the nuanced trade-offs necessary when balancing speed, accuracy, and resource allocation.

Figure 3 vividly illustrates the trade-off between latency and throughput across different algorithms, emphasizing the need to select solutions that align with specific application requirements in terms of speed and data volume management.

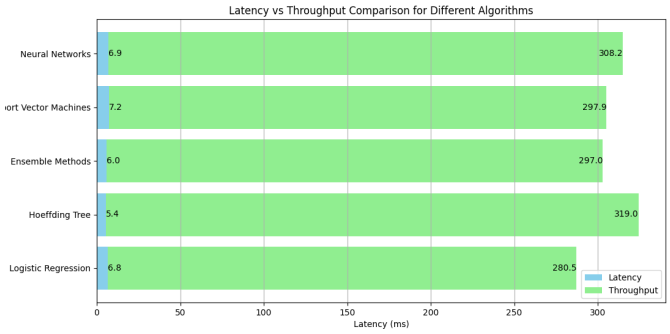

Figure 3: Latency vs. Throughput Comparison for Different Algorithms

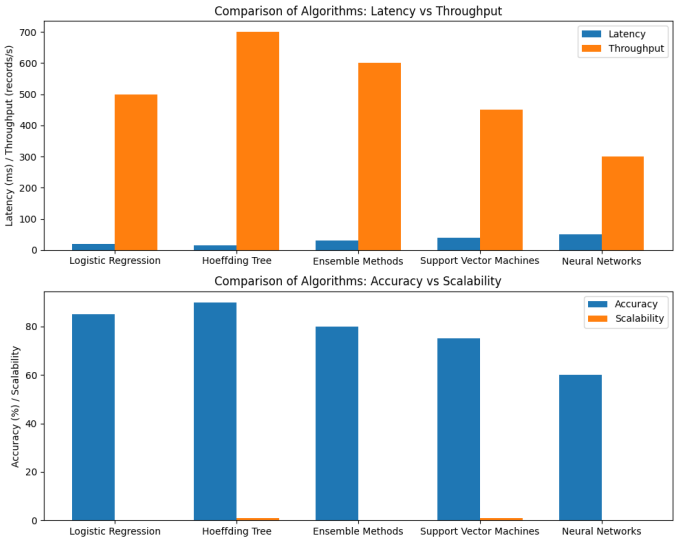

Figure 4: Accuracy vs. Scalability for Machine Learning Models

In Figure 4, we observe the relationship between accuracy and scalability among various machine learning models. Neural networks and ensemble methods, while offering high accuracy, necessitate significant resources to maintain scalability. Conversely, Hoeffding Trees exhibit a commendable balance, making them particularly effective in resource-constrained settings.

Ultimately, these results emphasize the criticality of contextual algorithm and architecture selection for real-time processing systems. The evaluation metrics serve as a valuable guide for decision-makers aiming to align system capabilities with business objectives, ensuring both efficacy and efficiency. As demonstrated, the interplay between algorithm performance and architectural design is fundamental in meeting the demands of real-time data analytics.

This comparative analysis, enriched by visualized metrics, lays bare the complexities and trade-offs involved in deploying advanced analytics systems. These insights provide a foundation for developing more tailored and effective solutions across various application domains.

7 Discussion

In our exploration of real-time data stream processing technologies, we delve into the operational performance of diverse algorithms and architectural designs. By scrutinizing metrics such as latency, throughput, accuracy, and scalability, this study illuminates pivotal considerations for selecting and implementing these solutions. This discussion seeks to unravel the significance of our findings, acknowledge inherent limitations, and chart a course for future inquiries.

7.1 Interpretation of Results

Our comparative analysis uncovers intriguing insights: Hoeffding Trees are particularly adept at maintaining low latency and high throughput, although they may fall short in accuracy when juxtaposed with ensemble methods. This presents a crucial trade-off for decision-makers in real-time applications who must weigh specific needs carefully. For example, in contexts like fraud detection and security monitoring, where rapid assessment is paramount to risk mitigation, the emphasis on minimizing latency becomes a decisive factor [15]. On the other hand, scenarios such as predictive maintenance—where the repercussions of false negatives are significant—might justify opting for ensemble methods despite their higher latency, thus ensuring greater precision in predictions.

Examining Lambda and Kappa architectures reveals distinct advantages; Lambda's dual-layer structure offers thorough data processing but can be complex to manage. In contrast, Kappa architecture provides a more streamlined approach that excels in environments where retrospective analysis is less critical, such as high-frequency e-commerce transactions [26]. These observations highlight the necessity for a nuanced understanding of both system requirements and operational contexts to harness effectively the advantages offered by real-time data processing technologies. The interplay between latency, scalabil-

ity, and accuracy significantly influences solution selection, demanding tailored consideration according to specific industry needs.

7.2 Limitations

The study's framework is inherently circumscribed by its reliance on controlled simulations and benchmarks, which introduces several limitations. A notable constraint is the assumption of ideal operational environments that often neglects the unpredictable variables encountered in real-world settings. Aspects like network bandwidth fluctuations, heterogeneous data formats, and system failures can profoundly impact performance metrics under genuine conditions [5].

Moreover, our implementation and evaluation of algorithms were based on a standardized set of input data configurations. In practice, variations in data characteristics such as volume, velocity, and variety could significantly alter processing efficiency and accuracy. Additionally, we did not comprehensively address deployment costs, maintenance overheads, or scalability constraints tied to infrastructure limitations—factors that are crucial in practical applications.

Furthermore, the focus on a selected group of algorithms and architectural models may limit the study's generalizability. While these techniques are widely recognized, emerging algorithms and innovative processing architectures continue to evolve, offering potential enhancements not captured within this analysis.

7.3 Implications for Practice and Future Research

The insights gleaned from our research bear substantial implications for practitioners engaged in real-time data analytics. Organizations must meticulously assess their requirements to ensure that technology choices align with performance objectives. For instance, sectors dealing with sensitive transactions such as finance and healthcare should prioritize accuracy and fault tolerance, even if it means embracing increased system complexity. Conversely, industries prioritizing speed, like digital marketing platforms, might benefit from adopting systems based on Kappa architecture for improved responsiveness [21].

Future research efforts should aim to explore advanced algorithms that balance speed with precision effectively. The potential of hybrid approaches, which amalgamate the strengths of various algorithms, could represent a promising direction. Additionally, investigating adaptive systems capable of dynamically adjusting parameters in response to data inflow and processing demands might yield further efficiencies.

The evolving realm of quantum computing also presents an exciting frontier for exploration. Its capacity to process vast datasets exponentially faster than traditional methods promises transformative opportunities for real-time analytics [20].

Moreover, with the growing emphasis on data privacy and security in real-time analytics systems, it is imperative that future studies address the integra-

tion of mechanisms ensuring compliance with regulations like GDPR and CCPA while maintaining analytical efficiency.

In conclusion, while our study sheds light on various algorithmic and architectural options within real-time data stream processing, embracing a more comprehensive view of real-world scenarios and emerging technologies will substantially broaden the scope of solutions in this dynamic field. This discussion underscores the importance of strategic alignment between technology capabilities and organizational goals, guiding future innovations in real-time data analytics.

References

[1] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. *SIAM International Conference on Data Mining*, 2010.

[2] Michael Stonebraker and Stanley Zdonik Cédric Plattner. The 8 requirements of real-time stream processing. *ACM SIGMOD Record*, 34:42–47, 2005.

[3] Arvind Arasu, Shivnath Babu, and Jennifer Widom. The cql continuous query language: Semantic foundations and query execution. *The VLDB Journal*, 15:121–142, 2006.

[4] S. Chandrasekaran and Others. Telegraphcq: Continuous dataflow processing for an uncertain world. *Conference: CIDR*, 2003.

[5] Ankit Toshniwal, Siddharth Taneja, Amit Shukla, Kavitha Ramasamy, Jason Schneider, Sujit Judah, Henry Ogilvie, Benjamin Schmidt, Sungwoo Lee, Jason Pauley, and et al. Storm@twitter. *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pages 147–156, 2014.

[6] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. Apache flink™: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 36:28–38, 2015.

[7] Lukasz Golab and M. Tamer Ozsu. Issues in data stream management. *ACM SIGMOD Record*, 32:5–14, 2003.

[8] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS '02)*, pages 1–16, 2002.

[9] Clyde L. Tucker. Window functions. *The Annals of Statistics*, 12(3):211–243, 1984.

[10] Arvind Arasu, Shivnath Babu, and Jennifer Widom. The cql continuous query language: Semantic foundations and query execution. *The VLDB Journal*, 15:121–142, 2006.

[11] Graham Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *Journal of Algorithms*, 55:58–75, 2005.

[12] David C. Luckham. The power of events. 2002.

[13] Gianpaolo Cugola and Alessandro Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, 44(3):15, 2012.

[14] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '00)*, pages 71–80, 2000.

[15] Joao Gama, Indrė Zliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4), 2010.

[16] Jay Kreps, Neha Narkhede, and Jun Rao. Kafka: A distributed messaging system for log processing. *Proceedings of the NetDB*, pages 1–7, 2011.

[17] Shahram Noghabi, Serdar Tasci, Zijie Qian, Yiyi Li, Jerry Liang, Jun Lee, Qi Zhang, Jun Li, Yaxing Zhang, Xiaowei Li, and et al. Samza: Stateful scalable stream processing at linkedin. *Proceedings of the VLDB Endowment*, 10(12):1634–1645, 2017.

[18] Raúl A. Fernandez. Towards a fault-tolerant big data technologies. *Computer Science - Research and Development*, 28:97–113, 2013.

[19] Paul C. Zikopoulos, Chris Eaton, D. deRoos, Tom Deutsch, and George Lapis. Understanding big data: Analytics for enterprise class hadoop and streaming data. 2011.

[20] Arimondo Scrivano. Quantum machine learning: Algorithms and applications. *N/A*, 2025.

[21] Scrivano Arimondo. A comparative study of recommender systems under big data constraints, 2025.

[22] Arimondo Scrivano. Fraud detection pipeline using machine learning: Methods, applications, and future directions, 2025.

[23] Scrivano Arimondo. A comparative study of classical and post-quantum cryptographic algorithms in the era of quantum computing. *arXiv preprint arXiv:2508.00832*, 2025.

[24] Arimondo Scrivano. Advances in indoor positioning systems: Integrating iot and machine learning for enhanced accuracy, 2025.

[25] Nathan Marz and James Warren. Big data: Principles and best practices of scalable realtime data systems, 2015.

[26] Jay Kreps. Questioning the lambda architecture. *O'Reilly Radar*, 2014.