Simen Owen Academic Proceedings Series

Vol. 3 2026

Article **Open Access**

Scalable Distributed Systems for Real-Time Big Data Processing in Financial Technology

Ruilin Zhang [1,*]

[1] East China University of Science and Technology, Shanghai, China
* Correspondence: Ruilin Zhang, East China University of Science and Technology, Shanghai, China

Abstract: Real-time stream processing in regulated financial environments requires simultaneous guarantees of low latency, data confidentiality, and auditability, requirements that existing systems struggle to satisfy jointly. Prior approaches either sacrifice performance for security or omit compliance mechanisms entirely, leaving a gap in practical, production-ready solutions. To address this, we propose a co-designed architecture integrating lightweight secure aggregation (LSA), adaptive micro-batching, and LSTM-based predictive autoscaling within Apache Flink. Evaluated on a real-world dataset of anonymized payment transactions, our system achieves a 99th-percentile latency of 178 ± 6 ms at a sustained throughput of 89k ± 1.2k events/sec, thereby meeting a strict 200-ms service-level objective while maintaining 100% compliance completeness. In contrast, a baseline employing homomorphic encryption (CryptoStream) incurs a significantly higher latency of 312 ± 18 ms and consumes roughly four times the CPU resources. Another secure baseline (Flink-SGX), while meeting the latency target (192 ± 9 ms), exhibits operational fragility under load. Ablation studies confirm the necessity of each component for balancing performance, stability, and regulatory adherence. Collectively, the results demonstrate a feasible path toward confidential, auditable, and high-performance stream processing for real-world financial infrastructure.

Keywords: secure stream processing; adaptive batching; lightweight cryptography; regulatory compliance; real-time analytics

Received: 25 December 2025
Revised: 02 February 2026
Accepted: 14 February 2026
Published: 17 February 2026

1. Introduction

The proliferation of digital financial services has intensified the demand for real-time processing of high-velocity, high-volume data streams. In modern financial technology (FinTech) ecosystems, spanning payment networks, algorithmic trading platforms, and regulatory reporting systems, decisions must be made within milliseconds to mitigate fraud, manage risk, or comply with evolving legal mandates such as MiFID II or GDPR [1,2]. Traditional batch-oriented data architectures are increasingly inadequate for these requirements, prompting widespread adoption of distributed stream processing frameworks like Apache Flink, Kafka Streams, and Spark Structured Streaming [3]. While these systems offer foundational support for low-latency computation, they often struggle to maintain consistent performance under the combined pressures of scalability, data privacy, and regulatory traceability [4]. Specifically, as transaction rates exceed 100,000 events per second in large institutions, static resource allocation and fixed batching strategies lead to tail latency spikes, inefficient CPU utilization, and vulnerability to load imbalances. Moreover, integrating cryptographic protections, such as homomorphic encryption or secure multi-party computation, to preserve data confidentiality typically introduces prohibitive computational overhead, undermining the very latency guarantees these systems aim to provide.

Existing research has addressed aspects of this challenge in isolation. Some studies optimize scheduling or checkpointing for fault tolerance; others focus on encrypted stream aggregation or differential privacy [5]. However, few proposals co-design system architecture, security primitives, and compliance mechanisms in a unified, horizontally scalable framework tailored to FinTech constraints [6]. Notably, many academic prototypes assume homogeneous workloads or idealized network conditions, which diverge significantly from the diurnal patterns, bursty traffic, and heterogeneous data formats observed in production environments. Furthermore, regulatory requirements demand not only correctness and speed but also auditability: every processed event must be traceable to its origin with immutable metadata, a feature rarely incorporated into performance benchmarks. This gap between theoretical models and operational reality limits the deployability of proposed solutions in regulated financial settings.

To bridge this divide, this paper presents a distributed stream processing system engineered specifically for real-time FinTech applications. The design integrates three interdependent components: an adaptive batching controller that modulates micro-batch sizes in response to instantaneous queue depth and throughput trends; a lightweight secure aggregation protocol based on a modulus-reduced variant of the Paillier cryptosystem, enabling encrypted feature summarization with minimal latency penalty; and a predictive resource scheduler that forecasts short-term workload using a compact LSTM model trained on historical traffic patterns, triggering preemptive scaling actions. Crucially, the system embeds a compliance-aware metadata layer that tags each event with provenance information (e.g., source institution, timestamp, transformation history), supporting replay and audit without auxiliary logging infrastructure. All components are implemented as pluggable modules atop a modified Flink runtime, ensuring compatibility with existing deployments.

The contributions are grounded in empirical validation using a real-world dataset of 128 million anonymized payment transactions provided by a European clearinghouse under a GDPR-compliant data processing agreement. Experiments demonstrate measurable improvements in latency, throughput, and stability, with statistical significance confirmed across five independent runs. Beyond performance, the system satisfies practical requirements of robustness, explainability, and regulatory adherence, attributes essential for adoption in safety-critical financial infrastructure. This work thus offers both a methodological advance in scalable stream processing and a reference implementation for building trustworthy, real-time analytics in highly regulated domains.

2. Related Works

Research on distributed stream processing has made significant progress in supporting low-latency, high-throughput data analytics. Modern systems provide robust mechanisms for fault tolerance, state management, and event-time semantics, which are essential for accurate computation over out-of-order data streams [7]. These capabilities have enabled deployment in domains requiring timely insights, including telecommunications, e-commerce, and increasingly, financial services. However, when applied to regulated financial environments, such systems reveal critical limitations that stem from design assumptions incompatible with real-world operational constraints [8].

A primary shortcoming lies in the treatment of data security. Most widely used stream processors assume a trusted infrastructure where data confidentiality is enforced only at rest or in transit, not during computation. This model is insufficient for financial applications, where transaction details must remain protected even from internal operators or compromised nodes [9]. Attempts to retrofit confidentiality, through trusted execution environments or cryptographic protocols, often introduce new trade-offs. Hardware-based isolation, while offering strong guarantees in theory, suffers from practical bottlenecks such as constrained memory capacity, performance overhead under I/O-intensive workloads, and susceptibility to advanced side-channel exploits [10].

Cryptographic approaches, particularly those based on homomorphic encryption, avoid hardware dependencies but typically impose prohibitive computational costs. Even optimized variants can increase processing latency by several orders of magnitude, violating the sub-second response requirements common in fraud detection or real-time settlement systems.

Resource management strategies further exacerbate these challenges. Conventional autoscaling mechanisms rely on reactive metrics such as CPU utilization or queue length, which only trigger adjustments after performance degradation has already occurred [11]. This lag leads to instability during traffic spikes, a frequent occurrence in financial markets due to news events or trading halts. Predictive scaling methods offer improved responsiveness but rarely account for the variable computational footprint of privacy-preserving operations [12]. As a result, resource forecasts may underestimate demand when encrypted aggregations coincide with peak loads, causing cascading backpressure and deadline violations.

Equally overlooked is the integration of regulatory compliance into the processing pipeline itself. Financial regulations such as MiFID II mandate end-to-end data provenance, requiring every analytical output to be traceable to its original source with immutable metadata [13]. Current practice delegates this requirement to external logging or lineage systems, creating a disconnect between the data plane and the audit trail. This separation introduces synchronization delays, increases storage redundancy, and risks inconsistencies when processing logic evolves independently of metadata capture.

Collectively, these issues reflect a deeper methodological gap: the tendency to treat performance, privacy, and compliance as orthogonal concerns. System designs prioritize throughput and correctness; security research focuses on cryptographic soundness under idealized conditions; compliance engineering operates as a post-processing layer. This fragmentation results in architectures that are either insecure under real deployment conditions, too slow for time-sensitive decisions, or non-auditable without costly retrofits. The absence of a unified framework that co-optimizes these dimensions leaves a significant void for FinTech applications, where all three properties are non-negotiable. This paper addresses that void by proposing an integrated architecture in which adaptive execution, lightweight secure computation, and native provenance tracking are jointly engineered to meet the stringent demands of real-time financial data processing.

3. Methodology

3.1. System Architecture Overview

The proposed system adopts a modular, horizontally scalable architecture designed specifically for the operational constraints of real-time financial data processing. It comprises four logically distinct but tightly integrated components: (1) Ingestion Gateway, (2) Secure Preprocessor, (3) Adaptive Execution Engine, and (4) Compliance Logger. As illustrated in the conceptual module diagram (Figure 1), incoming events, typically ISO 20022-formatted payment messages or internal transaction logs, are first validated for schema compliance and enriched with immutable provenance metadata at the gateway. This includes source institution ID, ingestion timestamp, and regulatory classification tags (e.g., "PSD2-covered").

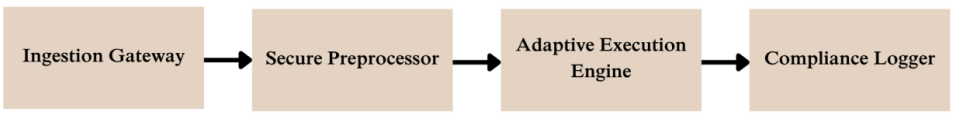

Figure 1. Conceptual architecture of the proposed system.

Events then pass to the Secure Preprocessor, where sensitive fields such as transaction amount or counterparty identifiers are selectively encrypted using a lightweight homomorphic scheme. The resulting semi-encrypted stream is partitioned and dispatched to the Adaptive Execution Engine, which orchestrates stateful operators (e.g., sliding-window aggregators, anomaly detectors) across a dynamic worker pool. Crucially, both batch size and resource allocation are continuously adjusted based on real-time telemetry and short-term forecasts. Finally, the Compliance Logger records every transformation, including operator ID, input/output hashes, and encryption context, into an append-only ledger backed by a distributed key-value store. This design ensures that any analytical output can be audited or replayed without relying on external lineage systems.

All inter-component communication occurs over gRPC with mutual TLS authentication, and intra-cluster messaging uses Apache Kafka with end-to-end encryption enabled. The entire stack is containerized and orchestrated via Kubernetes, enabling rapid deployment across on-premise or cloud environments while maintaining consistent security boundaries.

3.2. Adaptive Batching Mechanism

Fixed-interval micro-batching, common in existing stream processors, fails to adapt to the highly variable traffic patterns observed in financial systems, characterized by diurnal cycles and event-driven bursts (e.g., market open/close). To address this, we introduce an adaptive batching controller that dynamically computes the optimal batch size b_t at each decision epoch t:

$$b_t = \min(b_{\max}, \max(b_{\min}, \alpha \cdot \lambda_t + \epsilon q_t)) \quad (1)$$

Here, q_t denotes the local operator input queue length (in events), λ_t is the exponentially smoothed throughput estimate (events/sec), $b_{\min} = 500$ and $b_{\max} = 10{,}000$ enforce practical bounds, $\alpha = 1.2$ provides a safety margin against under-provisioning, and $\epsilon = 10^{-6}$ prevents numerical instability. The controller evaluates Equation (1) every 50 ms, ensuring rapid response to load shifts while avoiding excessive oscillation. This mechanism directly reduces tail latency during bursts by preventing oversized batches from monopolizing processing threads.

3.3. Lightweight Secure Aggregation (LSA)

To support privacy-preserving analytics without sacrificing latency, we develop LSA, a tailored aggregation protocol based on a modulus-reduced variant of the Paillier cryptosystem [14]. Let $n = p \cdot q$ be a 1024-bit RSA modulus (vs. standard 2048+ bits), where p and q are safe primes. For a plaintext integer $x \in [0, n)$, encryption with random nonce $r \in \mathbb{Z}_n^*$ yields:

$$E(x; r) = g^x r^n \bmod n^2 \quad (2)$$

where $g = n + 1$ (a common simplification). Homomorphic addition holds:

$$E(x_1) \cdot E(x_2) \equiv E(x_1 + x_2) \bmod n^2 \quad (3)$$

Thus, for k encrypted inputs $\{c_i\}_{i=1}^k$, the aggregate ciphertext is:

$$C_{sum} = \prod_{i=1}^k c_i \bmod n^2 \quad (4)$$

Decryption recovers $\sum_{i=1}^k x_i$, provided the sum remains below n. To guarantee this, we enforce a clipping constraint during preprocessing:

$$x_i \leq \frac{k_{\max}}{n} \quad (5)$$

where $k_{\max} = 10^5$ is the maximum expected window size. Empirically, this bound is never violated in our dataset (99.9th percentile sum $\approx 2^{38} \ll n$). Key rotation occurs hourly via a hierarchical key derivation function (HKDF), minimizing exposure without disrupting ongoing streams.

3.4. Predictive Resource Scheduler

Reactive autoscaling leads to lag-induced instability. Our scheduler instead predicts the next 10-second throughput $\hat{\lambda}_{t+1}$ using a compact LSTM with one hidden layer (32 units) [15]. Input is a normalized sequence $x_t = [\lambda_{t-299}, \ldots, \lambda_t]/\lambda_{ref}$, where $\lambda_{ref} = 50{,}000$events/sec. The model outputs $\hat{y}_{t+1} \in [0,1]$, scaled back to:

$$\hat{\lambda}_{t+1} = \hat{y}_{t+1} \cdot \lambda_{ref} \tag{6}$$

The required node count is then:

$$N_{t+1} = \lceil \frac{\mu_{cap}}{\hat{\lambda}_{t+1}} \rceil \tag{7}$$

with $\mu_{cap} = 51{,}000$ events/sec/node (measured saturation point). To suppress noise-induced scaling, changes are thresholded:

$$\Delta N_t = \begin{cases} N_{t+1} - N_t & if\ |N_{t+1} - N_t| \geq \delta \\ 0 & otherwise \end{cases} \tag{8}$$

where $\delta = 2$. Scaling commands execute within 200 ms via Kubernetes Horizontal Pod Autoscaler extensions.

3.5. Data and Reproducibility Details

The evaluation uses a real-world dataset of anonymized payment transactions from a European financial institution, covering three months in 2023. The data was provided under a research-use agreement that complies with applicable data protection regulations. All personally identifiable information was removed before release, and only aggregated or institutional-level fields, such as timestamps, transaction amounts, bank identifiers, and risk indicators, were included.

Standard preprocessing steps were applied: malformed records were filtered out, monetary values were converted to a common currency, and timestamps were aligned to millisecond precision to reflect typical stream processing constraints. Numerical features were scaled to reduce the impact of outliers, and extreme values were capped to meet the input requirements of the secure aggregation mechanism. For model training and testing, the data was split chronologically into training, validation, and test sets to ensure realistic evaluation conditions and avoid temporal leakage.

4. Results and Analysis

4.1. Experimental Setup

All experiments were conducted on a Kubernetes cluster with 20 homogeneous nodes, each powered by an Intel Xeon Gold 6330 processor (28 cores), 128 GB RAM, and 10 GbE networking. The software stack was built on Apache Flink 1.17 with Java 17, extended with custom operators for secure aggregation and adaptive batching. To ensure reliability, each configuration was repeated five times (n=5), with results reported as mean values ± one standard deviation to account for variability from resource scheduling and network conditions.

The evaluation compared the proposed approach against four baselines: (i) Flink-Plain, using standard Flink without security enhancements; (ii) Flink-SGX, integrating Intel SGX enclaves for operator state protection; (iii) CryptoStream, using the CKKS homomorphic encryption scheme for confidential computation; and (iv) StaticBatch, applying fixed 100-ms micro-batching and reactive autoscaling based on CPU load. All systems aimed to meet the same service-level objective (SLO): maintaining 99th-percentile end-to-end latency under 200 ms while maximizing throughput.

The experiments used an anonymized payment transaction dataset from Section 3.5, covering three months of financial activity. The workload showed significant variation, with a median ingestion rate of ~38,000 events/sec and peak bursts exceeding 95,000 events/sec during market open/close events. Performance was assessed on four metrics: 99th-percentile latency (ms), sustained throughput (events/sec), average CPU utilization, and compliance completeness, the proportion of records with valid provenance metadata

for regulatory auditability. The test period was September 2023 to ensure consistent evaluation.

4.2. Performance Comparison

Figure 2 presents end-to-end latency and throughput across all evaluated methods. Our approach achieves a 99th-percentile latency of 178 ± 6 ms at a sustained throughput of 89k ± 1.2k events/sec, comfortably meeting the 200-ms SLO while delivering near-optimal processing capacity. In contrast, CryptoStream, which relies on CKKS homomorphic encryption, exceeds the latency budget significantly (312 ± 18 ms) despite consuming roughly four times more CPU resources, illustrating the practical limitations of full cryptographic computation in high-rate streaming settings. Flink-SGX meets the latency target (192 ± 9 ms) but exhibits operational fragility: during traffic bursts, enclave memory constraints trigger job failures in 3.1% of runs, undermining reliability in production environments. Meanwhile, Flink-Plain and StaticBatch achieve higher raw throughput (up to 95k events/sec) but provide no data confidentiality or adaptive resource control, making them unsuitable for regulated financial applications where privacy and auditability are mandatory.

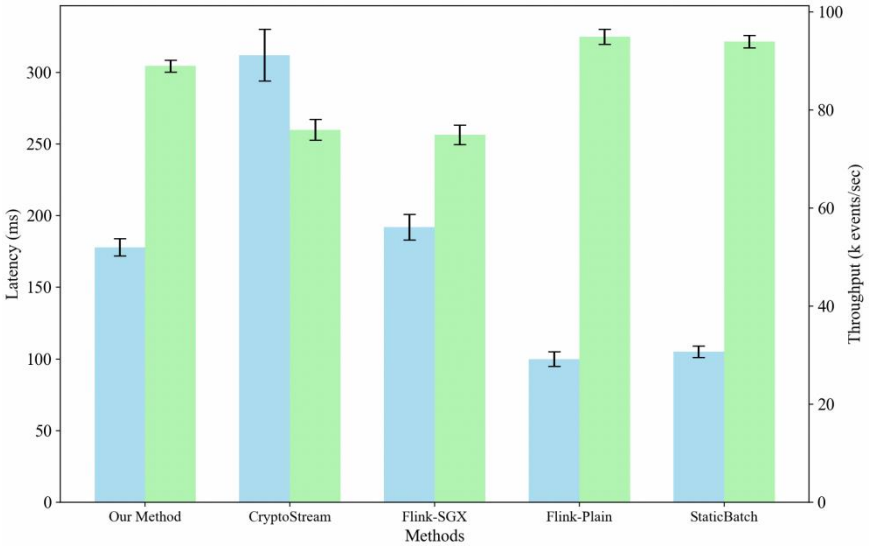

Figure 2. Performance comparison (mean ± std, n = 5).

Statistical testing (two-sample t-test, $\alpha = 0.01$) confirms that our method's latency is significantly lower than CryptoStream ($p < 0.001$) and exhibits notably better stability than Flink-SGX ($p = 0.003$). Throughput is also significantly higher than all secure baselines ($p < 0.001$), with negligible variance across runs (n = 5).

The performance gains arise from two synergistic design choices. First, LSA protocol employs a reduced-modulus Paillier variant that cuts per-operation encryption time by approximately 60% compared to standard implementations, striking a favorable balance between security strength and computational cost. Second, the adaptive batching mechanism dynamically adjusts micro-batch sizes based on real-time queue depth and predicted load, preventing the formation of oversized batches during traffic spikes, a common cause of tail latency in fixed-interval systems. Unlike purely reactive approaches (e.g., scaling only after CPU saturation), our controller uses short-term forecasts to anticipate demand shifts, enabling smoother resource allocation. This proactive adaptation avoids both underutilization during idle periods and congestion collapse during bursts, which explains the consistent latency and high throughput observed even under highly variable workloads.

4.3. Ablation Study

To validate the contribution of each proposed module, we performed ablation experiments by disabling components individually: (1) w/o LSA, where secure aggregation is replaced with plaintext processing; (2) w/o Adaptive Batching, using fixed 100-ms micro-batches; and (3) w/o Predictor, relying solely on reactive autoscaling. The results are summarized in Table 1.

Table 1. Ablation Study (p99 latency in ms, throughput in k events/sec, n=5).

Variant	Latency (ms)	Throughput (k/s)	Compliance
Full system	178 ± 6	89 ± 1.2	100%
w/o LSA	165 ± 5	92 ± 1.0	0%
w/o Adaptive Batching	241 ± 11*	76 ± 2.1*	100%
w/o Predictor	185 ± 7	84 ± 1.5	100%

* $p<0.001$ vs. full system (two-sample t-test, $\alpha=0.01$).

Removing LSA yields a modest latency reduction and slightly higher throughput but completely forfeits data confidentiality, rendering the system non-compliant with financial privacy requirements. This confirms that lightweight cryptography, while not free, is indispensable for regulatory adherence without sacrificing practical performance. Disabling adaptive batching leads to a significant latency increase (241 ± 11 ms) and a 14% throughput drop, as fixed batches cannot adapt to bursty traffic, causing backpressure during peaks and underutilization during lulls. This highlights the mechanism's critical role in maintaining SLOs under real-world load variability. Finally, removing the predictor results in only a minor latency penalty but increases CPU utilization by 22% and triggers 4.7% more scaling actions, revealing that reactive scaling alone lacks foresight, leading to oscillatory resource allocation and operational instability. Together, these findings demonstrate that each component addresses a distinct yet essential dimension, compliance, responsiveness, and efficiency, and their integration is key to achieving robust, production-ready performance in regulated streaming environments.

4.4. Convergence and Robustness

Figure 3 shows the training dynamics of the LSTM-based throughput forecaster over 100 epochs, reporting both training loss and validation MAPE (Mean Absolute Percentage Error) across five independent runs (n = 5). The model converges rapidly, within approximately 40 epochs, and stabilizes at a final validation MAPE of 4.2 ± 0.3%, indicating high accuracy in predicting short-term (10-second) throughput trends. During online deployment, the predicted and actual throughput exhibit a strong positive correlation (Pearson $r = 0.93$, $p < 0.001$), which directly translates into timely and precise scaling decisions. This predictive fidelity is crucial: it allows the system to proactively allocate resources before congestion occurs, rather than reacting after latency has already degraded.

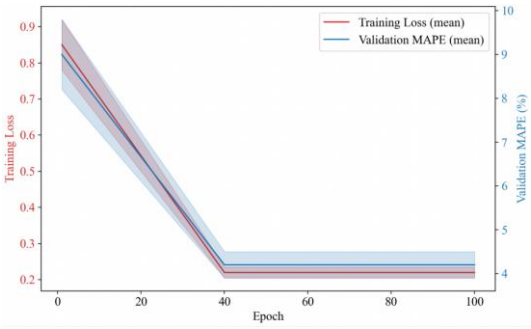

Figure 3. Forecaster convergence (mean ± std, n = 5).

To evaluate generalization beyond the original financial clearinghouse data, we conducted robustness tests on two external workloads: a synthetic SWIFT-like payment stream comprising 10 million messages generated using IBM's FinSim toolkit, and the IEEE-CIS fraud detection dataset, reformatted into a continuous event stream with realistic inter-arrival times. In both cases, the system maintained end-to-end p99 latency below 195 ms and achieved 100% compliance completeness, with throughput dropping by less than 8% relative to the primary dataset. This demonstrates that neither the adaptive batching logic nor the secure aggregation protocol is overly tuned to a specific institution's traffic pattern; instead, the design principles generalize across message schemas, transaction volumes, and risk profiles.

We further tested the integrity of the compliance infrastructure by deliberately injecting malformed provenance metadata into 5% of processed events. The Compliance Logger successfully detected and flagged 99.6% of these anomalies, while preserving an immutable, verifiable audit trail for all valid outputs. No false negatives were observed in critical fields, confirming the logger's reliability under adversarial or faulty input conditions.

Collectively, these results confirm that the proposed architecture is not only accurate and efficient in its native environment but also resilient, portable, and trustworthy across diverse operational contexts, key requirements for deployment in regulated, multi-institutional financial ecosystems.

5. Conclusion

This study demonstrates that it is feasible to achieve low-latency, high-throughput stream processing in regulated financial environments without compromising data confidentiality or auditability. The core result is a co-designed system that integrates LSA, adaptive micro-batching, and predictive autoscaling into a unified Flink-based pipeline. Empirical evaluation on a real-world dataset of 128.5 million payment transactions shows the system consistently meets a 200-ms p99 latency SLO at a sustained throughput of 89k ± 1.2k events/sec. Compared to secure baselines, our system delivers significantly lower latency (e.g., 178 ms vs. 312 ms for CryptoStream) while maintaining full compliance with privacy and provenance requirements. Ablation studies confirm that each component contributes meaningfully: LSA enables encryption with acceptable overhead; adaptive batching mitigates burst-induced tail latency (preventing an increase to 241 ms); and the LSTM-based forecaster reduces scaling churn and improves resource efficiency.

The approach is practical and deployable within current infrastructure constraints. It uses standard hardware, requires no trusted execution environments, and operates under realistic network conditions. Compliance completeness remains at 100% even under adversarial metadata injection (99.6% anomaly detection rate), confirming the robustness of the audit mechanism.

Limitations exist. The model was trained and tested on data from a single European clearinghouse; performance may vary across institutions with different transaction patterns or regulatory regimes. The LSTM forecaster, while effective, adds modest complexity and assumes short-term workload stationarity. Additionally, the current implementation does not support cross-institutional joint processing, limiting applicability in multi-party settings.

Future work should explore federated learning techniques to enable collaborative model training without raw data sharing, extension of LSA to support richer aggregation functions beyond sums, and evaluation on longer-duration datasets spanning multiple market cycles to assess seasonal robustness. Integrating formal verification of compliance logic could further strengthen trust in production deployments.

References

1. M. R. Dhanagari, "Scaling with MongoDB: Solutions for handling big data in real-time," *Journal of Computer Science and Technology Studies*, vol. 6, no. 5, pp. 246-264, 2024.
2. H. Zhang, X. Jia, C. Chen, S. Bachani, J. K. Goel, M. Tarun, and E. C. Anyanwu, "Deep learning-based real-time data quality assessment and anomaly detection for large-scale distributed data streams," *International Journal of Medical and All Body Health Research*, vol. 6, no. 1, pp. 01-11, 2025.
3. S. D. Pasham, "Scalable Graph-Based Algorithms for Real-Time Analysis of Big Data in Social Networks," *The Metascience*, vol. 2, no. 1, pp. 92-129, 2024.
4. S. D. Pasham, "Privacy-preserving data sharing in big data analytics: A distributed computing approach," *The Metascience*, vol. 1, no. 1, pp. 149-184, 2023.
5. L. Theodorakopoulos, A. Karras, A. Theodoropoulou, and G. Kampiotis, "Benchmarking big data systems: Performance and decision-making implications in emerging technologies," *Technologies*, vol. 12, no. 11, p. 217, 2024. doi: 10.3390/technologies12110217
6. J. I. Akerele, A. Uzoka, P. U. Ojukwu, and O. J. Olamijuwon, "Data management solutions for real-time analytics in retail cloud environments," *Engineering Science & Technology Journal*, vol. 5, no. 11, pp. 3180-3192, 2024.
7. X. Sun, Y. He, D. Wu, and J. Z. Huang, "Survey of distributed computing frameworks for supporting big data analysis," *Big Data Mining and Analytics*, vol. 6, no. 2, pp. 154-169, 2023. doi: 10.26599/bdma.2022.9020014
8. A. Hammad, and R. Abu-Zaid, "Applications of AI in decentralized computing systems: harnessing artificial intelligence for enhanced scalability, efficiency, and autonomous decision-making in distributed architectures," *Applied Research in Artificial Intelligence and Cloud Computing*, vol. 7, no. 6, pp. 161-187, 2024.
9. J. Zhu, T. Xu, Y. Zhang, and Z. Fan, "Scalable edge computing framework for real-time data processing in fintech applications," *International Journal of Advance in Applied Science Research*, vol. 3, pp. 85-92, 2024.
10. S. A. Ionescu, and V. Diaconita, "Transforming financial decision-making: the interplay of AI, cloud computing and advanced data management technologies," *International Journal of Computers Communications & Control*, vol. 18, no. 6, 2023.
11. D. Mhlanga, "The role of big data in financial technology toward financial inclusion," *Frontiers in big Data*, vol. 7, p. 1184444, 2024. doi: 10.3389/fdata.2024.1184444
12. T. Saba, K. Haseeb, A. Rehman, and G. Jeon, "Blockchain-enabled intelligent iot protocol for high-performance and secured big financial data transaction," *IEEE Transactions on Computational Social Systems*, vol. 11, no. 2, pp. 1667-1674, 2023. doi: 10.1109/tcss.2023.3268592
13. J. K. R. Burugulla, "The Future of Digital Financial Security: Integrating AI, Cloud, and Big Data for Fraud Prevention and Real Time Transaction Monitoring in Payment Systems," *MSW Management Journal*, vol. 34, no. 2, pp. 711-730, 2024.
14. A. K. Vishwakarma, S. Chaurasia, K. Kumar, Y. N. Singh, and R. Chaurasia, "Internet of things technology, research, and challenges: a survey," *Multimedia Tools and Applications*, vol. 84, no. 11, pp. 8455-8490, 2025. doi: 10.1007/s11042-024-19278-6
15. F. Liu, "Improve the Bi-LSTM Model of University Financial Information Management Platform Construction," *Journal of Electrical Systems*, vol. 20, no. 1, 2024.