JOURNAL OF INTERNATIONAL CRISIS AND RISK COMMUNICATION RESEARCH
ISSN: 2576-0017
2026, VOL 9, NO 1

Self-Healing AI-Native Real-Time Data Pipelines: Autonomous Resilience For Large-Scale Streaming Systems

Yogesh Pugazhendhi Duraisamy Rajamani

Independent researcher, USA

Abstract

In large streaming platforms today, there are common operational issues, such as data drift, throughput degradation, partition imbalance, and cascading failures, that impact availability and performance. Existing monitoring and rule-based automatic remediation solutions are unsuitable for workloads with millisecond-level latency and high availability needs. This article introduces a fully self-healing AI-native real-time data pipeline that integrates machine learning into the control plane of the streaming platform. It presents an end-to-end architecture that leverages graph neural networks and transformers for hybrid anomaly detection, LSTM-based predictive fault modeling, and reinforcement learning-based agents that autonomously select the best remediation policy (e.g., dynamic resource scaling, partition rebalancing, and dataflow rerouting). The framework implements continuous healing based on the detect-diagnose-predict-decide-act-verify-learn loop. Evaluating the framework with synthetic and real-world high-throughput streaming workloads shows improvements in downtime, latency, fault domains, and resource utilization to establish a new model of autonomous stream processing infrastructures that can continue to operate mission-critical workloads in cloud, hybrid, and edge environments.

Keywords: Real-Time Streaming Systems, Self-Healing Architectures, Reinforcement Learning, Anomaly Detection, Autonomous Fault Mitigation.

I. Introduction

1.1 Emergence of High-Velocity Data Ecosystems and Infrastructure Complications

Digital transformation initiatives across industries have fueled dramatic and continuous growth in the data streaming from sensor networks, monitoring infrastructure, and behavior tracking interfaces. Today, compute infrastructure must support the streaming of this data at unprecedented data rates and with the latency on the order of microseconds and zero-downtime availability. Time-sensitive applications require instantaneous collection of data, on-the-fly transformation pipelines of data, and real-time analytical interpretation to create algorithms capable of reacting quickly enough. Instant processing architectures are also needed in Internet of Things deployments as data velocity and volume storage requirements accelerate [1]. Operational challenges in production-grade streaming environments pose a threat to the reliability guarantees and throughput provided. Statistical pattern evolution introduces uncertainty in the data characteristics over diverse time windows. Fluctuations in the computation requirements lead to provisioning challenges not addressed by static resource allocation strategies. Various pipeline bottlenecks introduce backpressure and reduce throughput. In distributed systems, hardware failures such as crashes of servers and partitions between networked data centers can create further problems. Sub-

millisecond latency requirements are so strict that there is virtually no headroom for recovery time or latency of any kind.

1.2 Inadequacies of Conventional Oversight and Operator-Driven Remediation

Customary monitoring and manual processes have fundamental deficiencies when working in enterprise-scale systems. Rule-based alert systems generate too many notifications for operations teams to handle, and automatically initiated responses to incidents introduce delays between detection and recovery. Static configuration methods cannot adapt to the continually changing workloads, while administrators with limited human resources cannot manage the increasing complexity of the streaming service's topology graph. This and the time it takes for the administrator functioning in the loop to act upon an anomaly can lead to cascading failures, data loss events, and outages that obstruct key business operational goals. Khattach et al. show how integrated architectural solutions with machine learning components can benefit real-time analytics and predictive maintenance functions for IoT platforms and highlight the shortcomings of existing monitoring models [2].

1.3 Research Goals and Key Innovations

In this context, the project anticipated the embodiment of autonomous resilience capabilities into the operational layer of streaming infrastructures by embedding computational learning models inside their orchestration control planes, enabling automated anomaly detection, failure prediction, and fault tolerance, without human intervention, setting the stage for continuous self-optimization and fault-tolerant capabilities for edge infrastructures. An important result of the project was the definition of an integrated framework for anomaly detection, predictive analytics, and reinforcement learning-based decision-making mechanisms for streaming coordination subsystems. The proposed framework is a layered architecture for hybrid detection engines comprising graph neural network topologies and transformer-based resources for identifying complex failure patterns, as well as a long short-term memory forecasting module designed to enable predictive resource management through early fault detection. An optimization-based decision module can automatically select the best corrective actions from defined strategy spaces. Experiments under high workload conditions show the benefits of improvements in system availability, latency variability, and resource utilization.

II. Related Work and Background

2.1 Evolution of Autonomous Systems and AIOps

From scripted automation through to autonomous computing systems, over many decades, AIOps practices in their current form have emerged as the technical and operational complexity of distributed computing environments has advanced to a point where organizations are looking to use machine intelligence to reduce their reliance upon manual management. Initial implementations of the model have been in narrow use cases such as log aggregation analysis and event correlation, later evolving into a more enterprise-wide operational governance framework. This evolution signals a growing recognition that customary approaches to infrastructure management do not scale to the complexity and pace of operation of modern systems [3]. Modern autonomous architectures that include the perceptive, diagnostic, planning, and executive layers create feedback control circuits that allow for continuous adaptation to the environment. These technological changes have shifted the infrastructure management perspective of enterprises from a reactive incident response approach to being proactive in improving the performance of their organizations.

2.2 Existing Approaches to Fault Tolerance

Policy-driven systems are the basic resilience mechanisms. They express operational knowledge with predefined policy rules to specify the behavior of infrastructure during irregular conditions. Such mechanisms typically consist of triggering recovery actions when monitored properties cross some operational threshold. Many rule-based monitoring systems have been built on top of such mechanisms, specifying a set of conditional rules with branching pathways and proposed responses to observed behavior. These approaches are deterministic, observably transitively closed, and dependably localized to the state but are brittle to new manifestations of failure and continue to require manual calibration as the

underlying infrastructure evolves. Fixed autoscaling implementations, specifically in cloud computing environments, are a solution to the resource allocation problem via preconfigured elasticity policies triggered by metric threshold crossings. These solutions lack workload awareness and cannot forecast workload demand peaks and drops. Segregated machine learning anomaly identification is a recent subset of these solutions based on probabilistic and deep learning algorithms to detect deviations from the nominal operational signature. Such techniques have greater sensitivity to small disturbances but are normally used as separate monitors rather than feedback control loops, resulting in a discontinuity between sensing and correction.

Table I: Comparison of Fault Tolerance Approaches [3, 4]

Approach Category	Detection Method	Response Mechanism	Adaptability	Workload Awareness	Integration Level
Policy-Based Systems	Threshold Monitoring	Predefined Rules	Static	Low	Isolated
Rule-Driven Monitoring	Conditional Logic	Alert Generation	Limited	Low	Isolated
Static Autoscaling	Metric Thresholds	Resource Provisioning	Fixed Policies	Minimal	Platform-Level
Isolated ML Anomaly Detection	Statistical Models	External Alerting	Moderate	Medium	Monitoring-Only
AI-Native Self-Healing	Hybrid GNN + Transformer	Autonomous Remediation	Dynamic	High	Embedded Control Plane

2.3 Gap Analysis and Streaming Platform Fundamentals

Architecturally, a challenge in existing fault tolerance frameworks is that they do not consistently combine identification, prediction, and automated remediation. Existing frameworks tackle operational gaps separately. These schemes lack integrated intelligence at all stages of fault management, including surveillance, forecasting, decision synthesis, and execution. As a result, fragmented responses are enforced, with potential inefficient resource allocation leading to over-engineering or under-engineering. Streaming platforms are systems that continuously collect, aggregate, and process streamed data. For example, Apache Kafka implements distributed event streaming based on partitioned topic logs and a consumer group coordination protocol. Apache Flink provides stateful stream features using dataflow programming abstractions with a temporal event model and exactly-once delivery guarantees. Spark Structured Streaming extends batch execution programming models to micro-batch and progressive computation [4]. These platforms are formulated as substrate infrastructures under which intelligence layers achieve autonomous fault governance objectives, requiring computational learning to be deeply embedded within stream processing execution topologies and orchestration control logic.

III. System Architecture and Design

3.1 Five-Layer Architectural Framework

The architecture discussed in the paper implements a five-layer, hierarchical organization of functional strata that support self-contained fault management. The lowest functional stratum, the Streaming Ingestion Layer, is responsible for the acquisition of high-rate events from multiple event sources and the persistence of message repositories that stabilize incoming streams at the event gate. The Distributed Processing Layer runs parallelized computation frameworks to execute transformation functions, stateful logic flows, and analytics on partitioned data constructs, while the AI Control Plane implements

computational intelligence within infrastructure coordination substrates. In addition, prognostic models, irregularity detection algorithms, and decision synthesis engines continuously assess the health of system components. The Self-Healing Orchestration Engine translates diagnostics into reaction plans that coordinate actions such as workload migration, partition equilibration, and repartitioning of resources, thus enabling the system to act on its own. The Serving and Observability Layer provides universal transparency across the architecture layers by aggregating all the telemetry information, creating historical analysis of it, and giving proof-of-presence records for concurrent observation and back-in-time forensics [6]. This multi-level decomposition of all other layers ensures modularization and functional separation, while keeping the intelligence subsystems close-coupled to the conduits of operational execution.

Table 2: Five-Layer Architecture Components and Functions [5, 6]

Layer	Primary Functions	Key Technologies	Operational Scope	Integration Points
Streaming Ingestion	Event capture, message buffering, durable queuing	Kafka, Pulsar	Data acquisition	Processing Layer
Distributed Processing	Parallel computation, stateful operations, transformations	Flink, Spark Streaming	Data transformation	AI Control Plane
AI Control Plane	Anomaly detection, fault prediction, decision optimization	GNN, Transformer, LSTM, RL	Intelligence layer	Orchestration Engine
Self-Healing Orchestration	Task migration, partition rebalancing, and resource allocation	Kubernetes, YARN	Automated remediation	All layers
Serving and Observability	Telemetry collection, trend analysis, and audit logging	Prometheus, Grafana	Monitoring and visibility	All layers

3.2 Integration of ML Models into Pipeline Control Plane

Machine learning algorithms are part of the governance layer that extends surveillance beyond external APIs. Prognostic algorithms are embedded into stream processing job supervisors and cluster administrators, where they consume active streams of telemetry data from operational computation tasks and infrastructure vitality monitors. Detecting irregularities with Irregularity identification networks involves taking into account execution topology metrics, resource consumption, and dataflow characteristics to detect irregularities before they trigger failures. The projection modules provide predictions based on historical performance sequences of resource requirements, queue build-up, and future constraints over finite intervals. Decision synthesis components take the outputs of diagnostic components and operational constraints to specify remediation strategies that optimize system objectives subject to the associated disruption costs. This situating of the decision synthesis process avoids the latency associated with the externally situated monitoring infrastructure, leading to sub-second response times between irregularity detection and remediation. Model execution occurs within the same operational context as the corresponding stream processing, using the same compute resources and physically co-located state repositories.

3.3 Design Principles for Autonomous Operation

Autonomous operations models contrast with human-controlled systems based on the general principles of perpetual observation mandates, which involve the instrumentation of all architectural layers to log resource expenditures, dataflow rates, processing latencies, and frequencies of anomalous events.

Whereas reactive incident engagement is triggered on service degradation, anticipatory cognizance principles use projection models to chart trajectories of system behavior and point out incipient anomalies before service disruption occurs. Closed-circuit automation addresses the steps of interruption identification, interpretation, disruption decision-making, and disruption removal. By closing the loop in this way, closed-circuit automation allows the fast closure of response loops and the containment of disruptions. Acceptable degradation enables infrastructure to remain usable at reduced capability upon component failures through prioritized processing and dynamic quality of service. Learning law, such as feedback circuits, merges performance knowledge into models, creating channels, continually optimize both prediction quality and determination reliability based on observed remediation outcomes amid various failure conditions.

3.4 Deployment Considerations Across Computing Environments

The deployment architecture designation has a pronounced influence over the customizability and operational characteristics of autonomous systems targeting cloud, hybrid, and edge implementations. Cloud-indigenous deployments utilize elastic infrastructural provisioning and curatorial service frameworks to help scale resources and reduce operational footprint through provider-managed governance planes [5]. Hybrid architectures have processes split between premises-based and cloud-based data facilities, balancing data sovereignty with on-demand scalability at the cost of latency and synchronization. Edge computing architectures colocate processing resources with the data generation site to reduce ingestion latencies and transmission overhead at the cost of scarcity and possibly intermittent connectivity [5]. These deployment archetypes impose different requirements on the autonomous administration mechanisms concerning their resource allocation strategies, disruption detection thresholds, and remediation strategies. In cloud environments, aggressive autoscaling approaches and low provisioning times may be possible, while in edge environments, resource conservation and the possibility of limited functionality may become important in resource-constrained deployments. Hybrid architectures, therefore, require cross-environment synchronization mechanics to allow consistent behaviors across different infrastructures and network conditions.

IV. AI-Native Components and Methodology

4.1 Anomaly Detection Engine

The anomaly detection engine employs a graph neural network and transformer attention protocols to detect anomalous behavior patterns across distributed streaming workloads. A graph neural network is used to construct computational topology as an arrangement of interconnected vertices, effectively capturing the spatial relationships between computational processing operators, data partitions, and resource provisioning conditions. Transformer architectures can expand this spatial representation with temporal attention constructs along with sequentially arranged metric progressions over observation timelines, diagnosing subtle deviations in execution attributes that are precursors of eventual system failures. This allows structural relationships of directed acyclic graph execution configurations and temporal evolutions of metrics to be assessed jointly [7]. The detection system is supplied continuously with telemetry currents, including operator throughput velocities, buffer saturation magnitudes, checkpoint finalization intervals, and inter-task transmission delays. Feature derivation conduits recast telemetry currents into standardized encodings suitably processed by the neural network. Attention weight allocations pinpoint computational subgraphs showing deviant operational behavior. Within a detection framework, probabilistic irregularity quantifications offer metrics of deviation severity. Subsequent decision infrastructures use them to prioritize remediation activities according to the magnitude of envisioned consequences.

4.2 Predictive Fault Modeling

Forecasting faults involves leveraging different long short-term memory network configurations to predict the expected system degradation across various dimensions of the system. LSTM-based forecast modules analyze long time series data of queue build-ups, processing delay distributions, and backpressure distributions to forecast the next state of the infrastructure across customizable model

horizons. These recurrent architectures contain internal memory states to encode the long-term temporal patterns observed in the queueing process, allowing for the identification of degradation patterns when observing the process over long horizons. Each projection module encodes a unique disruption type, with dedicated networks for each of the configurations of queue growth rates (under insufficient processing capacity), latency growth probabilities (under resource competition), and backpressure patterns (under constraint). The node and partition disruption projection is improved with additional feature streams containing hardware power measures, network connection assessments, and workload distribution properties. Point extrapolations are accompanied by confidence boundaries. By running decision infrastructures on top of the model, remediation efforts can be scaled by the degree of uncertainty and the severity of consequences.

4.3 Reinforcement Learning Decision Engine

The reinforcement learning decision apparatus represents operational governance as a sequential decision problem and allows autonomous agents to perform interventions that realize response options as part of a long-term plan pertaining to the infrastructure. The response options are identified as: actions that intervene through horizontal expansion as part of the deployment of processing capacity to change the operator's concurrency levels, actions that migrate processed workloads to redistribute computations across executive units, actions that reroute data traffic to change the structure of message paths, and actions that balance partitions to redistribute parallel processing workloads. State encodings in stateful RL techniques compress present infrastructure measurements, contemporary irregularity detection conclusions, and operational workload specifications into compressed feature matrices encoding operational circumstances [8]. The reward architecture construction assimilates numerous conflicting targets through weighted amalgamations of processing delay sanctions, infrastructure constancy metrics quantified through measurement fluctuation, and resource utilization expenditures mirroring platform costs. Policy networks enact agent determination operations by mapping the observed conditions into distributions over intervention probabilities, balancing the benefits of exploiting productive policies with exploring alternate interventions. Off-policy learning protocols allow continuous policy improvement from historical operational records, improving the quality of decisions through experiences gathered from a range of disruptions and associated workload levels.

4.4 Autonomous Healing Loop

In order to autonomously ease closed-circuit operation for vitality stewardship of the streaming platform, the autonomous healing circuit creates a seven-stage operational loop: Detection, which continuously analyzes streaming telemetry data to identify anomalies between the measurements and the derived baselines, or deviations resulting in anomalous patterns being identified. Diagnosis detected faults or anomalies and their potential causes by studying the spreading of a disruption across the network in computational topology and past data of incidents over a period. Prediction considered the likely evolution of the infrastructure given the conditions at a given moment in time through degradation timelines and expected disruptions in the processing phases. The Decide stage synthesizes diagnosis and prognosis results to select the optimum remediation strategy from a set of interventions, considering their costs and expected benefits [8]. The Act stage executes the selected remediation strategy using orchestration interfaces by requesting resource provisioning, migrating workloads, and changing configurations of components of the distributed platform. The Verify stage monitors operational infrastructure, tracking the value of remedial efforts through metrics governing stabilization and irregularity resolution. The Learn stage incorporates operational outcomes via conduits of model refinement, adjusting detection sensitivity thresholds, prediction accuracy boundaries, and determination policy coefficients depending on remedial impact observed, improving operational infrastructure performance via autonomously evolving capabilities.

Table 3: AI-Native Component Specifications [7, 8]

Component	Architecture Type	Input Features	Output Products	Operational Phase	Temporal Scope
Anomaly Detection Engine	Hybrid GNN + Transformer	Operator throughput, buffer occupancy, checkpoint duration, and communication latency	Probabilistic anomaly scores, subgraph highlights	Detection, Diagnosis	Real-time
Predictive Fault Modeling	LSTM Networks	Queue growth, latency distributions, backpressure patterns, and hardware health	Future state projections, confidence intervals	Prediction	Short- to medium-term
RL Decision Engine	Policy Network	System metrics, anomaly outputs, workload characteristics	Action probability distributions, intervention selection	Decide, Act	Real-time
Autonomous Healing Loop	Seven-Phase Cycle	Multi-source telemetry, historical outcomes	Remediation directives, performance feedback	All phases	Continuous

V. Experimental Evaluation and Discussion

5.1 Experimental Setup and Evaluation Metrics

The experimental validation apparatus applies exhaustive measurement protocols to quantify the effective potency of the autonomous healing infrastructure over a range of key operational dimensions. Downtime reduction measurements quantify the aggregate duration of service unavailability experienced under baseline conditions with manual intervention, versus deployments with autonomous healing, across the various dimensions. Latency stability tests compute the variance of the processing delay, measuring the capability of the infrastructure to keep its response features unchanged despite disturbances to the underlying platform and load conditions. Fault containment potency measure is based on the measurement of spatial fault boundaries, that is, the capability of the infrastructure to localize the disturbances within computing domains. The resource usage productivity metric evaluates resource usage properties of a computational resource provisioning, comparing the self-determined decisions to provision to a maximum throughput-to-cost ratio against static provisioning and human-controlled orchestration of provisioning. In evaluation frameworks for such metrics, synthetic workload generators are used to produce controllable event streams. Production traces with specific workload characteristics can then be reproduced. Measurement capture describes delay allocation telemetry at millisecond granularity, asset consumption trajectories at the second interval, and event-magnitude processing throughput statistics across a distributed computing assembly as clearly as possible.

Table 4: Experimental Evaluation Metrics and Workload Configurations [1, 2, 9, 10]

Evaluation Metric	Measurement Unit	Baseline Comparison	Evaluation Period	Collection Granularity
Downtime Reduction	Service unavailability intervals	Manual intervention	Extended operations	Event-level
Latency Stability	Processing delay variance	Static configuration	Continuous monitoring	Millisecond-resolution
Fault Containment	Spatial propagation boundaries	Traditional isolation	Per-incident	Subgraph-level
Resource Utilization	Throughput-to-cost ratio	Fixed provisioning	Operational lifetime	Second-interval

5.2 Performance Analysis Across High-Throughput Workloads

Performance tests use heterogeneous high-throughput workload configurations across operational and stress conditions. Steady state tests examine constant event rates near peak capacity boundaries and infrastructure behavior under sustained resource contention conditions without inducing artificial downtime. Burst tests repeatedly exceed baseline load to probe the ability to reduce demand surges and quickly provision additional resources. Degrade tests do not involve an artificial load spike. Instead, they simulate the progressive exhaustion of the platform's capacity by gradually increasing latency and adding artificial interruptions to evaluate model accuracy and preemptive/predictive actions. Heterogeneous workload scenarios combine streaming analytics, stateful aggregation, and complex event processing (CEP) operations in one joint execution network, and experiment with various computational and resource consumption footprints. A multi-tenant arrangement tests isolated application deployments across a shared infrastructure, evaluating autonomous control in a mixed service contention and separation environment. A geographic allocation arrangement tests how processing assemblies are split across multiple data center regions, subjecting telecommunication to a range of latencies and enforcing distributed healing protocol cooperation.

5.3 Limitations and Challenges

Despite its efficacy, several drawbacks still need to be addressed before the autonomous healing system can be widely applicable. Model drift in dynamic workloads is a critical challenge since streaming applications can vary considerably during their life cycles, impacting the ability to precisely recognize and predict anomalies, especially when the training data is different from the active environment. In practice, refinement protocols for the perpetual model incur a computational cost that needs to be managed and orchestrated effectively, so as not to interfere with primary processing workloads. Unseen failure configuration sensitivity can be viewed as a supervised learning problem, as novel failure classifications with no prior knowledge in the longitudinal training data are often undetected or off-target in remediation interventions. Confidence intervals become wider, and performance may degrade during instability signatures for which there is no experience. Further, growing the reinforcement learning agent would require more exploration time to develop a more complex policy, during which the agent may take actions that lead to poorer results for the infrastructure. Successfully balancing exploration needs with invariance requires advanced curriculum learning and safety constraints on exploration, at least in the initial phases where these invariances are being cultured.

5.4 Future Research Directions

Future augmentations will propel autonomous streaming platforms further; digital twin simulations can be used for protected policy analysis or to evaluate disruption situations without having any impact on the production infrastructure [9]. Additionally, virtual twins, including the characteristics of the production platform, can provide accelerated cultivation schedules, synthetic disruption for policy testing at scale, and hypothetical tests for optimal bandwidth orchestration. Federated learning for distributed conduits can enforce privacy preservation and geographic information sovereignty mandates by allowing the training of a distributed model across organizational boundaries while preserving sensitive operational telemetry

[10]. In the context of multiple organizations deploying similar streaming services, distributed learning conventions enable a collaborative approach to improving irregularity detection architectures and decision-making algorithms while keeping data custody and competitive integrity locally. Multi-agent reinforcement learning approaches extend single-agent decision-making tools to multi-agent synchronous optimization in a microservice architecture. Possible application domains include emergent synchronization behavior assisting to achieve global infrastructure goals via localized decision-making, exploring alternative models to classical, centralized governance approaches for tackling large-scale distributed settings. Other areas of research to explore include explainable AI, creating operational transparency, transfer learning for quick adaptation across diverse, streaming infrastructures, or quantum-inspired optimization methods for complex resource allocation processes.

Conclusion

For large-scale streaming infrastructures where throughput and complexity grow rapidly, monitoring-based repair as a default response is not viable. Article proposes a self-healing AI-native architecture for autonomously resilient streaming infrastructures, which leverages embedded computational intelligence for anomaly detection, predictive fault modeling, and reinforcement learning-based decision synthesis in the streaming platform control plane. The architecture comprises five layers: ingestion layer, distributed computing layer, AI control plane, self-healing orchestration layer, and observability layer, guided by a seven-phase autonomous healing loop. Hybrid graph neural networks and transformers are used for complex irregularity detection across computer topologies, and LSTM-based forecasting modules are used to proactively anticipate and address potential disruptions. Practical reinforcement learning agents automatically learn which remediation action from scaling, migration, rerouting, and rebalancing to take to move the system towards its goals, given its operational history. Experiments with different high-throughput workload scenarios show improvements in availability, latency variability, fault isolation, and resource utilization. This article lays a foundation for next-generation streaming systems that enable workload automation in cloud, hybrid, and edge environments, thereby realizing fully autonomous, always-on data infrastructure ecosystems that continuously self-adapt, self-optimize, and self-repair throughout their operational lifetime.

References

[1] P. Rajasekar, et al., "Real-time Stream Processing in IoT Environments," in 2024 Ninth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), Chennai, India, June 28, 2024, pp. 1-6. [Online]. Available: https://ieeexplore.ieee.org/document/10568668

[2] Ouiam Khattach, et al., "End-to-End Architecture for Real-Time IoT Analytics and Predictive Maintenance Using Stream Processing and ML Pipelines," Sensors, vol. 25, no. 9, p. 2945, May 7, 2025. [Online]. Available: https://www.mdpi.com/1424-8220/25/9/2945

[3] Kyriakos M. Deliparaschos, et al., "Facilitating Autonomous Systems with AI-Based Fault Tolerance and Computational Resource Economy," Electronics, vol. 9, no. 5, p. 788, May 10, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/5/788

[4] Williams Sarah, "A Performance Benchmark of Apache Flink, Apache Spark, and Kafka Streams in Real-Time ML Pipelines," ResearchGate Preprint, February 2, 2025. [Online]. Available: https://www.researchgate.net/publication/391644409

[5] Francesco Cosimo Andriulo, et al., "Edge Computing and Cloud Computing for Internet of Things: A Review," Informatics, vol. 11, no. 4, p. 71, September 30, 2024. [Online]. Available: https://www.mdpi.com/2227-9709/11/4/71

[6] Alexandros Papanikolaou, et al., "Introducing Responsibly Self-Healing into the Incident Management Lifecycle," in Proceedings of the 16th International Conference on Pervasive Technologies Related to Assistive Environments (PETRA '23), July 2023. [Online]. Available: https://dl.acm.org/doi/fullHtml/10.1145/3594806.3594837

[7] Qian Yang, et al., "Graph Transformer Network Incorporating Sparse Representation for Multivariate Time Series Anomaly Detection," Electronics, vol. 13, no. 11, p. 2032, May 23, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/11/2032
[8] Mirco Theile, et al., "Position Paper: Deep Reinforcement Learning for Real-Time Resource Management," Real-Time Systems, June 5, 2025. [Online]. Available: https://link.springer.com/article/10.1007/s11241-025-09443-x
[9] Han Li and Tianzhen Hong, "A digital twin platform for building performance monitoring and optimization: Performance simulation and case studies," Building Simulation, June 13, 2025. [Online]. Available: https://link.springer.com/article/10.1007/s12273-025-1290-2
[10] Bassel Soudan, et al., "Scalability and performance evaluation of federated learning frameworks: a comparative analysis," International Journal of Machine Learning and Cybernetics, February 1, 2025. [Online]. Available: https://link.springer.com/article/10.1007/s13042-024-02453-4