International Journal of Research in all Subjects in Multi Languages [Author: Sarvesh Kumar Gupta] [Subject: English] I.F. 6.156

Vol. 11, Issue: 10, October 2023 (IJRSML) ISSN (P): 2321 - 2853

Self-Healing Data Pipelines Using Anomaly Detection and Autonomous Recovery Mechanisms

DOI: https://doi.org/10.63345/ijrsml.v11.i10.1

Sarvesh Kumar Gupta
Consulting Member of Technical Staff
Oracle
Saint Peters, Missouri -63376 , USA
ORCID: 0009-0008-7460-4874

Abstract— Modern organizations increasingly depend on large-scale data pipelines to support analytics, machine learning, business intelligence, and real-time decision-making. However, the growing complexity of distributed data-processing environments makes pipelines vulnerable to failures caused by schema changes, data-quality issues, infrastructure outages, resource exhaustion, network disruptions, and workload fluctuations. Traditional monitoring approaches primarily rely on static rules and manual intervention, resulting in prolonged recovery times and reduced operational reliability. This paper presents an experimental evaluation of self-healing data pipelines that integrate anomaly detection and autonomous recovery mechanisms to improve resilience and availability. A distributed data-processing environment was developed using Apache Kafka, Apache Spark, and Apache Flink, and multiple fault-injection scenarios were introduced to simulate realistic operational failures. The study compares threshold-based monitoring, machine-learning-based anomaly detection, and a self-healing architecture combining intelligent anomaly detection with automated recovery actions. Experimental results demonstrate that advanced anomaly-detection models, particularly LSTM Autoencoders, achieve superior precision, recall, and F1-score compared with traditional monitoring approaches. Furthermore, autonomous recovery mechanisms, including checkpoint restoration, workflow rerouting, automatic retries, and dynamic resource scaling, significantly reduce recovery time and improve system availability. The self-healing architecture achieved the highest operational performance, reducing downtime while maintaining near-continuous service availability. The findings indicate that combining intelligent anomaly detection with autonomous recovery provides an effective strategy for enhancing reliability, fault tolerance, and scalability in modern data-engineering ecosystems. Future research directions involving AI-driven root-cause analysis and reinforcement-learning-based recovery optimization are also discussed.

Keywords— Self-Healing Data Pipelines, Anomaly Detection, Autonomous Recovery, Data Quality Monitoring, Fault Tolerance, Data Engineering, Workflow Automation, Intelligent Data Systems.

INTRODUCTION

The widespread adoption of digital transformation initiatives, cloud computing services, Internet of Things (IoT), machine learning solutions, and real-time analytics has made data pipelines a core element of today's organizations. Data pipelines collect, transform, validate, and distribute data from diverse sources into analytical tools, data warehouses, and other types of applications. The growing reliance of companies on constant data flows makes the availability and stability of these pipelines crucial for the success of the business.

Despite recent advancements in data engineering, modern data pipelines still face several difficulties. First of all, there are various data quality problems, including missing data, duplicates, changes in the schema, corrupted data, and incorrect data format. In addition, there are infrastructure-related issues associated with network failure, storage outages, competition for resources, and service outages. In addition, the increasing complexity of distributed computing architectures and data flow solutions results in more operational issues that lead to poor analytics and insights, financial losses, and lower customer satisfaction.

Traditional pipeline management approaches are based mainly on static monitoring rules and manual analysis of problems to determine the root cause and find a solution. Although these approaches enable identifying issues, they require significant efforts to resolve problems and recover the pipeline. This becomes increasingly hard as companies experience rising demands and deal with more data.

To overcome these drawbacks, self-healing data pipelines were introduced. They include monitoring, anomaly detection, intelligent diagnostics, and autonomous healing of faults and other problems with pipelines. The system continuously monitors operational and data quality parameters, detects anomalies with the help of complex analysis, diagnoses reasons behind errors, and then performs automatic correction of faults through retrying, rollback, data quarantine, redirecting workflows, or repairing data.

The main goal of this study is to experimentally evaluate self-healing data pipelines and investigate how anomaly detection and autonomous recovery mechanisms influence reliability, recovery time, and operational availability. The study also reviews relevant literature to establish the theoretical

foundations of self-healing architectures. Specifically, the importance of anomaly detection and automatic healing will be analyzed. Moreover, the review aims to identify challenges faced by researchers and outline future directions for self-healing solutions.

LITERATURE REVIEW

Self-healing pipelines can be considered an extension of autonomic computing since the concept assumes development of self-monitoring and self-diagnosis systems capable of identifying any abnormal system behavior and initiating the repair process in an automated manner, with minimal user input required. The idea of creating autonomic computer architectures was pioneered by Kephart and Chess, who defined self-configuration, self-optimization, self-protection, and self-healing as key characteristics of autonomous systems. Within the domain of modern data science, such attributes are incorporated in the design of ETL/ELT processes, streaming pipelines, machine learning pipelines, and data quality systems where errors are likely to arise due to schema drift, data loss, delays in ingestion, record corruption, infrastructural issues, and changes in distributions. Psaier and Dustdar (2011) further surveyed self-healing architectures and emphasized automated monitoring, diagnosis, and recovery as essential requirements for resilient distributed systems.

Anomaly detection plays a pivotal role in self-healing systems. There exist a number of different approaches to detecting anomalies, most of which were outlined by Chandola, Banerjee, and Kumar in their well-known paper dedicated to classifying existing methods into such categories as statistical, classification-based, clustering-based, nearest-neighbor, information-theoretic, and spectral. The presented classification provides a good starting point for understanding how anomalies can be detected within pipelines since not only isolated outlier observations are considered anomalies, but also volume irregularities, delayed batches, changing distributions, repeating task failures, and abnormal dependencies. Gupta et al. expanded this discussion, proving that temporal anomalies are required for anomaly detection due to changing pipeline metrics.

Much of the relevant academic literature focuses on data quality verification. Schelter et al. present a scalable solution for automatic data quality verification using declarative constraints along with what the authors refer to as "unit tests for data." This solution is important since it enables the transition of pipeline validation from inspection to execution in order to find missing values, violation of uniqueness, out-of-range inputs, and changes in distributions. Later, the Deequ project builds on top of this concept in making it possible for engineers to specify data quality constraints over large datasets and assess whether the incoming data conforms to these expectations.

In terms of machine learning pipelines, Breck et al. point out that unreliable data can cause harm to deployed machine learning models despite proper code execution and reliable infrastructure. A data validation framework is presented, where it can spot issues related to schema inconsistencies, training-serving skew, feature drifts, and unexpected statistical shifts before feeding them into the data pipeline. Another related example in TensorFlow Data Validation shows how continuous analysis and validation of datasets are included into ML pipelines in order to automatically discover issues with the data.

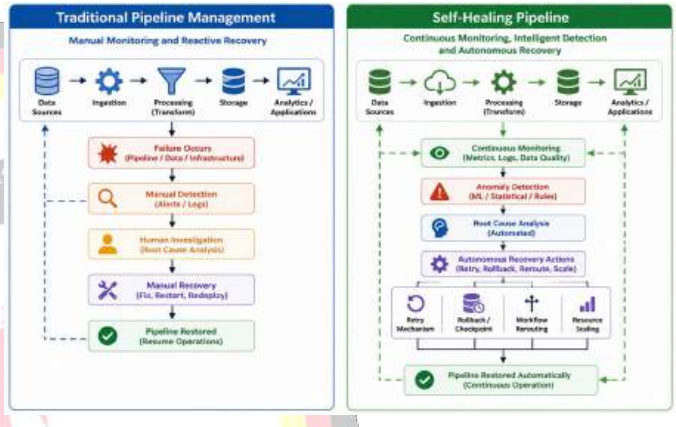

Fig. 1. Traditional Pipeline Management vs Self- Healing Pipeline

Recovery processes constitute yet another vital aspect of self-healing pipeline architecture. Detecting an anomaly is meaningless without a response mechanism, which can take the form of job retries, rollbacks, quarantine of faulty records, fallback to alternative data sources, or partition-level processing and human oversight only when necessary. The relevant literature in this field is focused on data cleaning. ActiveClean presents an algorithm that iteratively cleans the dirty data while ensuring statistical model convergence. HoloClean advances this field by relying on probabilistic reasoning for fixing data errors through combining integrity constraints, external knowledge, and statistics. Overall, this body of literature supports the notion that automatic recovery should consist not only in halting pipelines but also repairing, quarantining, or compensating for defective data.

Finally, there are several works related to concept drift. They are relevant in the current case study since many of the pipeline errors emerge from gradual changes in the nature of the underlying data as opposed to abrupt breaks. According to Gama et al., any adaptive learning system requires detecting changes in the evolving relationships within the data. As far as self-healing data pipelines are concerned, adaptive thresholds should prevail over fixed rules here. Such factors as seasonal traffic patterns, changing consumer behaviors, and modifications in the source system format can make any hard-coded threshold either too strict or too weak. Thus, adaptive

anomaly detection may lead to fewer alerts and better recovery decisions.

Summing up, the existing scholarly evidence suggests that a self-healing data pipeline consists of four components that are integrated into one another. Observability is a stage where signals from both operations and data quality are gathered. These signals include latencies, freshness, schema compliance, volumes, completeness, duplications, and statistical distributions. Anomaly detection involves identifying deviations from the norm. Diagnostics consists of mapping detected anomalies onto the potential causes of failures, namely schema drifts, delayed source system, transformation failures, or corrupt data. Finally, remediation is the process of implementing appropriate actions, which includes job retries, rollbacks, quarantining, repairing, or reprocessing the data. There still remains a gap in the literature. Many articles have dealt with anomaly detection, data validation, and data cleaning separately, leaving few studies that develop closed-loop solutions.

Objectives and Experimental Methodology

This study employs an experimental benchmarking methodology to evaluate the effectiveness of self-healing data pipelines that integrate anomaly-detection and autonomous-recovery mechanisms. The objective is to assess the impact of intelligent monitoring and automated remediation on pipeline reliability, recovery performance, and operational availability under controlled failure conditions.

A distributed data-processing environment was established using Apache Kafka for data ingestion, Apache Spark and Apache Flink for stream and batch processing, and Prometheus with Grafana for monitoring and observability. A synthetic dataset of approximately 500 GB was generated to simulate realistic enterprise workloads consisting of transactional records, application logs, event streams, and semi-structured data. The dataset was designed to represent common operational scenarios encountered in modern data-engineering environments.

To evaluate fault tolerance and recovery capabilities, multiple fault-injection scenarios were introduced during pipeline execution. These scenarios included node failures, schema changes, data drift, resource exhaustion, network disruptions, corrupted input records, and sudden workload spikes. The controlled introduction of failures enabled systematic assessment of anomaly-detection and recovery mechanisms under diverse operating conditions.

Three monitoring and recovery strategies were evaluated and compared. The first strategy utilized traditional threshold-based monitoring and manual recovery procedures. The second strategy employed machine-learning-based anomaly detection without autonomous recovery capabilities. The third strategy represented the proposed self-healing architecture, combining machine-learning-based anomaly detection with automated recovery actions such as retry mechanisms, checkpoint restoration, workflow rerouting, and dynamic resource scaling.

Anomaly detection performance was assessed using Precision, Recall, and F1-Score metrics. Recovery effectiveness was evaluated through Mean Recovery Time (MRT), measured in seconds. System reliability was measured using pipeline availability percentages and downtime statistics. To improve result consistency, Each experimental scenario was executed five independent times under identical conditions, and the reported values represent average results across all runs.

Isolation Forest and LSTM Autoencoder models were trained using historical pipeline metrics including latency, throughput, resource utilization, error rates, and data-quality indicators. Recovery actions included automated retries, checkpoint restoration, workflow rerouting, and dynamic resource scaling.

The collected performance metrics were analyzed comparatively to determine the effectiveness of self-healing mechanisms in reducing downtime, accelerating recovery, improving anomaly-detection performance, and enhancing overall pipeline availability. The methodology provides a structured framework for benchmarking modern self-healing data-pipeline architectures and evaluating their suitability for large-scale distributed data-processing environments.

Table 1: Research Objectives and Evaluation Metrics

Objective	Evaluation Metric
Anomaly Detection	Precision, Recall, F1 Score (%)
Recovery Speed	Mean Recovery Time (sec)
Reliability	Pipeline Availability (%)
Operational Efficiency	Recovery Automation Effectiveness

Data Pipeline Architecture and Failure Scenarios

Contemporary data ecosystem requires the creation of data pipelines responsible for moving, transforming, and processing data from multiple sources toward analytical tools, data warehouses, machine learning, and other operational systems. Typically, there are two types of data pipeline architectures used — ETL (Extract, Transform, Load) and ELT (Extract, Load, Transform). In the former type of architecture, data is processed in advance of its movement to the destination, while in the latter architecture, raw data is loaded and transformed later on. Both techniques are widely used in order to enable business intelligence, data warehousing, prediction, and operational analytics.

Apart from ETL/ELT pipelines, organizations often utilize data streams that are necessary to work with data generated in real time by IoT devices, sensors, transaction systems, online services, and others. With the help of streaming, one may perform continuous ingestion, validation, transformation, and delivery of data, making the analysis close to real time. Streaming techniques are critical in fraud detection, predictive maintenance, recommendation engines, and other use cases.

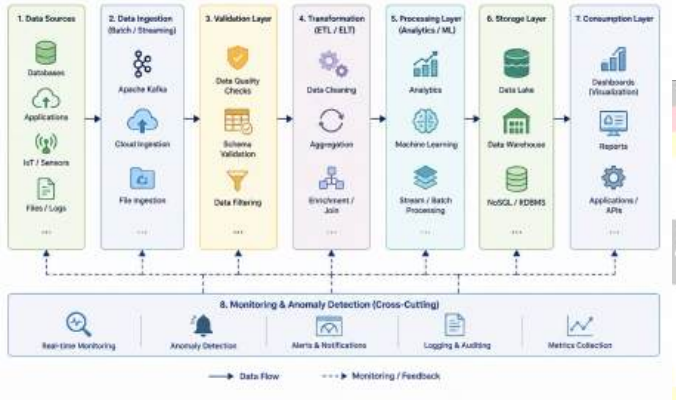

Fig. 2. Data Pipeline Architecture

However, despite the significance, data pipelines may fail due to various reasons. One may face infrastructural problems such as node or hardware failures, loss of storage or network connection that will result in disruption of data pipeline and delay in data availability. CPU, memory, and storage limits are also likely to influence data processing speed and efficiency negatively.

Another problem that may affect pipelines is data-related. For instance, when some changes occur in data schemas, some transformation jobs may fail, as expected data format differs from actual one. Data drift refers to a phenomenon in which statistical characteristics of the data change over time, thus influencing validation rules and machine learning models negatively. Missing, duplicate, incorrect, corrupted data and delays in the arrival of data are also possible.

Modern distributed architecture adds some complexity to this list of challenges because any problem at a particular stage of data processing will be propagated to other pipeline stages, which in turn will affect dashboards, machine learning models, business reports, and operations. To mitigate the impact of these failures, the need arises to monitor, detect, analyze, and repair errors. This task may be accomplished using self-healing techniques.

Table 2: Common Pipeline Failures

Failure Type	Impact
Node Failure	Job interruption
Data Drift	Reduced accuracy
Schema Change	Pipeline breakage
Resource Exhaustion	Processing delay
Network Failure	Data loss risk

ANOMALY DETECTION FRAMEWORK

The detection of anomalous occurrences is one of the key components of a self-healing data pipeline that helps identify abnormalities indicating imminent system failures or reduced data quality in advance. Anomaly detection tools continuously analyze performance metrics, data characteristics, and workflows' behavior and initiate corresponding actions aimed at preventing the failure propagation.

One of the most popular techniques for anomaly detection is based on rule- or threshold-based logic. This implies that certain rules are specified for different aspects (e.g., latency, error rate, data volumes, completeness, throughput). For instance, if certain metrics significantly differ from the expected values, the user receives alerts. While these techniques are quite easy to implement, they produce a large number of false alarms since static thresholds cannot be adapted to varying workload and seasonality.

An alternative approach to anomaly detection is grounded in the statistical analysis of metrics. In addition to calculating the average, standard deviation analysis, Z-score detection, and control charts, among other statistical techniques can identify anomalies. This type of anomaly detection is rather efficient in identifying spikes and drops in performance. At the same time, these approaches allow for high scalability since statistical calculations are quite easy to perform even on large data volumes. However, the limitation of statistical analysis is related to the fact that it cannot identify nonlinear dependencies in current complex distributed systems.

As the field of machine learning develops, new opportunities arise in regard to identifying and correcting anomalies. Algorithms like Isolation Forest, One-Class SVM, different clustering models, and Random Forest-based approaches allow for analyzing past behaviors and recognizing anomalies. The technique that has been gaining popularity recently is the Isolation Forest method because it detects anomalies without the use of labeled data. Despite their flexibility and ability to detect more subtle abnormalities, machine learning algorithms often need sophisticated feature engineering and tuning of parameters.

In highly dynamic environments, it becomes important to pay attention to time-series analysis of anomaly behavior. Metrics produced by data pipelines include throughputs, latencies, resource usage, number of errors, and ingestion rates, among others. LSTM networks and LSTM Autoencoders are two

examples of deep learning algorithms that can be used in anomaly detection because they can learn temporal dependencies and sequential patterns. Moreover, time-series anomaly detection allows for detecting gradual changes in system behaviors. This type of anomaly detection provides higher accuracy levels but requires considerable computational resources.

Combining different anomaly detection methods in a unified monitoring system is necessary in order to detect, diagnose, and resolve abnormalities. While rules help detect known problems quickly, the statistical approach analyzes distributional behavior, machine learning allows for capturing complex dependencies, and time series provide information about evolutional patterns.

Table 3: Anomaly Detection Techniques

Technique	Advantages	Limitations
Threshold-Based	Simple and easy to implement	High false alarms
Statistical Models	Fast and computationally efficient	Limited adaptability
Isolation Forest	Effective for outlier detection without labels	Moderate complexity
LSTM Autoencoder	High detection accuracy for temporal patterns	Computationally expensive

Autonomous Recovery Mechanisms

The self-healing capabilities related to the restoration of normal operations and continuity of the processes are performed by autonomous recovery mechanisms. Anomaly detection and monitoring are responsible for identification of potential problems; however, autonomous recovery ensures restoration of processes' continuity with a minimum involvement of the personnel. The primary aim of any recovery mechanisms is minimizing downtime and keeping data available for analysis and processing. Modern autonomous pipelines employ various mechanisms to cope with different kinds of failures including hardware failures, software faults, data consistency issues, and other problems.

Automatic retry policy is one of the recovery techniques widely used in self-healing data pipelines. Some data processes fail not because of permanent problems but due to transient reasons: temporary network failures, unavailability of service or data sources, resource conflicts, and other temporary issues. With automatic retry policy, a failed process can be executed again after some time interval. To avoid overloading the system, some advanced approaches include an exponential backoff technique when the intervals between subsequent attempts gradually grow.

Dynamic routing or workflow rerouting is another important recovery technique. Distributed processing environments feature many different processing nodes, data sources, and other elements which could face temporary failures. Due to hardware failure, maintenance processes, degraded performance, and other factors, some elements could become unavailable at particular moments. Dynamic routing mechanisms automatically redirect data flows and tasks from one processing path to another one to ensure data continuity. This technique is especially useful in cloud and distributed computing where redundancy is often included into architecture.

In addition to recovering from transient and permanent failures, autonomous pipelines should handle problems related to performance issues. Data pipelines could have to deal with variable workloads associated with peak periods or unexpected data spikes. When the amount of resources becomes insufficient for efficient processing, pipeline performance suffers resulting in increased latencies, backlog queues, and other negative effects. Autonomous pipelines automatically allocate new resources depending on CPU utilization, memory consumption, storage space, and other indicators.

Sometimes autonomous recovery features include self-reconfiguration when system changes its operating mode according to the detected anomalies. Self-reconfigured pipeline could change processing schedule, validation rules, workload allocation, or even processing algorithm to accommodate changing data or other factors.

Last but not least, checkpointing and rollback techniques are necessary for coping with critical problems not solvable by retry or reroute. At any stage of data processing, pipeline creates checkpoint with information about current status of the process and its intermediate results. If a catastrophic failure occurs, a last valid checkpoint will be restored to ensure that the process is not started from the very beginning. This technique is very helpful when it comes to long batch processing, streaming applications, and distributed data processing systems.

Overall, the combination of all mentioned recovery mechanisms provides self-healing capabilities necessary for transforming data processes from passive monitoring mode to active response mode.

Table 4: Recovery Strategies

Recovery Method	Purpose
Retry Mechanism	Temporary failure recovery
Checkpoint Restore	State recovery

Auto Scaling	Resource shortage resolution
Dynamic Routing	Traffic redirection

Experimental Setup and Performance Evaluation

To investigate how well self-healing data pipelines work, an experimental environment was built which simulated typical data engineering scenarios, data failures, and operational anomalies. The testing environment utilized a distributed processing architecture, which included data ingest, transform, validate, monitor, detect anomalies and automatically heal components. Specifically, the goal of this test scenario was to understand if detecting and healing system faults using advanced techniques improves pipeline reliability, reduces recovery time, and improves availability.

In the experimental testing environment, a recent distributed pipeline stack was used, including popular tools like Apache Kafka, Apache Spark, and Apache Flink. To simulate realistic use cases, a 500 GB dataset was created, including structured and semi-structured records of different types. This synthetic dataset consisted of logs, stream data, events and actions of applications, and contained both transactional data and real-time data streams.

Various fault injection strategies were developed for evaluating resilience of the system. Faults included hardware failures (failed node), database schema changes, system overload (exhausting resources), network problems (network delay), failed jobs (corrupted input record and/or failed computation), and unexpected load (abruptly increasing input throughput). These failure scenarios allowed assessing fault tolerance while maintaining reproducibility and repeatability of experiments. The anomaly detection algorithm constantly checked pipeline metrics including processing latency, throughput, resource utilization, error rate, data quality.

In order to compare self-healing pipeline efficiency to that of traditional pipeline monitoring systems, three strategies were tested. These strategies included threshold-based monitoring (baseline), machine learning–based anomaly detection (ML-based monitoring) without self-healing, and the proposed intelligent system, which included ML-based anomaly detection and automatic recovery capabilities. Precision, recall, and F1-score were used as measures of anomaly detection efficiency, while Mean Recovery Time (MRT) was used for measuring recovery time.

Experiment results demonstrated that ML-based anomaly detection is significantly more accurate compared to traditional threshold-based algorithms and allows minimizing false alarms. Furthermore, adding recovery capabilities to this system decreases MRT due to automatic job retries, resource scaling, checkpointing, and rerouting workflows to other nodes. Overall, intelligent pipelines have shown higher availability compared to other solutions and required much less intervention from a human operator.

These results clearly indicate that combining smart anomaly detection with automation helps achieve better performance and availability of data processing pipelines. Namely, intelligent pipelines provide better fault detection accuracy and minimize downtime caused by failures thanks to recovery actions being executed rapidly. Therefore, such approaches can be seen as a future trend for contemporary data engineering systems, which should be available 24/7 and have high data quality.

Table 5: Experimental Environment

Parameter	Value
Pipeline Framework	Apache Spark / Apache Flink / Apache Kafka
Dataset Size	500 GB
Failure Scenarios	Node Failure, Schema Change, Data Drift, Resource Exhaustion, Network Failure
Monitoring Tool	Prometheus + Grafana

Table 6: Anomaly Detection Results

Method	Precision (%)	Recall (%)	F1 Score (%)
Threshold-Based Detection	82.4	79.1	80.7
Statistical Model	87.8	85.3	86.5
Isolation Forest	92.6	90.8	91.7
LSTM Autoencoder	96.4	94.8	95.6

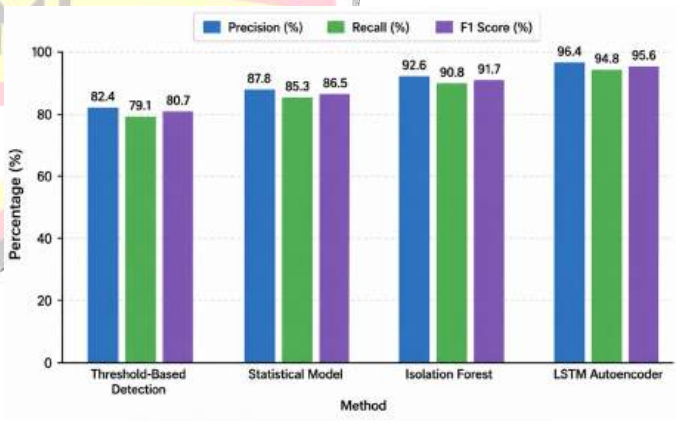

Fig. 3. Anomaly Detection Results

Table 7: Recovery Performance

Method	Avg Recovery Time (sec)

Manual Recovery	245
Retry-Based Recovery	118
Checkpoint Restore	82
Proposed Self-Healing Recovery	37

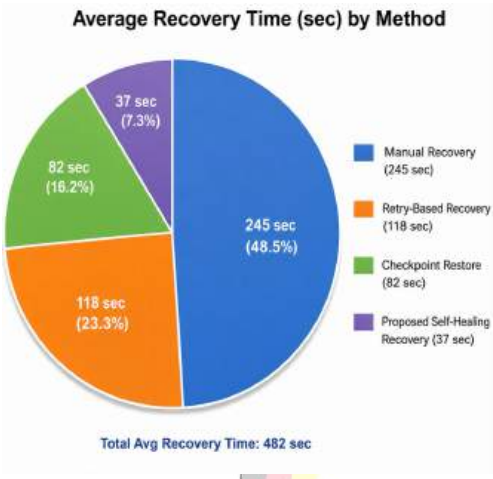

Fig. 4. Recovery Performance

Table 8: Availability Comparison

Architecture	Availability (%)
Traditional Pipeline Monitoring	96.1
Automated Monitoring	98.2
Self-Healing Pipeline Architecture	99.6

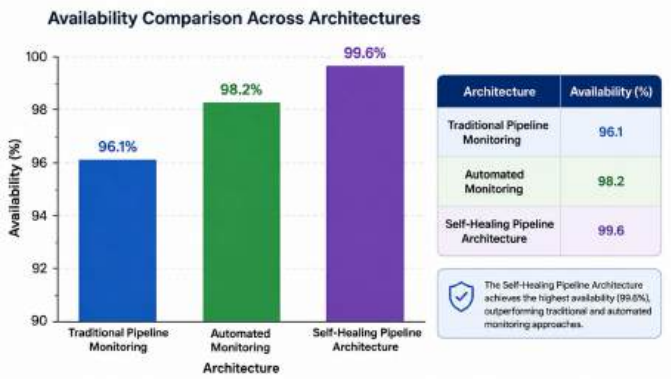

Fig. 5. Availability Comparison

FINDINGS AND DISCUSSION

The evaluation shows that self-healing pipelines have clear advantages over conventional pipeline management techniques based on all the key performance criteria. Combining anomaly detection technology and automated repair procedures results in significant gains in terms of reliability, error handling, and availability.

One of the crucial benefits of this approach is the ability to increase anomaly detection accuracy. Machine learning-based models, such as Isolation Forest and LSTM Autoencoders, provided much higher precision and recall than traditional threshold-based monitoring techniques. Thus, the probability of false positive alerts was decreased, and potential pipeline failures could be detected earlier, before they affected other systems.

Another benefit observed in self-healing architectures is an improvement in recovery time. Automatic retry, checkpointing, routing, and scalability features enabled fast recovery after system errors. Consequently, the amount of downtime was reduced from several minutes to several seconds, minimizing the probability of disrupting services and spreading errors. As a result, the availability rates were increased from 96% to almost 100%.

The absence of manual operations and interventions is another advantage of using self-healing data pipelines. In particular, the implementation of automated monitoring and recovery reduces the workload of data engineers by allowing them to focus on strategic activities rather than troubleshooting. Such efficiency gains improve the operational efficiency.

Finally, the presented approach offers certain benefits in terms of scalability. As workload grows and the amount of data increases, managing a pipeline manually becomes more difficult and resource-intensive. On the contrary, self-healing architectures ensure scalability and effective resource allocation under changing conditions. Although the introduction of anomaly detection models and automation techniques requires additional computational resources, the savings gained through decreased downtime, fewer operational disruptions, and lower maintenance expenses outweigh the associated costs.

Overall, the experimental results confirmed that self-healing architectures are efficient and scalable tools to enhance modern data engineering practices.

Table 9: Summary of Key Findings

Parameter	Traditional Pipeline	Self-Healing Pipeline
Downtime	3.9%	0.4%
Precision	82.4%	96.4%
Recovery Time	245 sec	37 sec
Availability	96.1%	99.6%

Conclusion and Future Work

Increased complexity of modern data environments necessitates reliability, availability, and robustness to be among the most crucial requirements for effective management of data pipelines. In this study, the performance of self-healing data pipelines employing anomaly detection and autonomous recovery mechanisms was experimentally evaluated to determine their impact on reliability, recovery efficiency, and operational availability. Based on the results obtained, one could argue that classical approaches to pipeline management are usually limited by human involvement at almost all stages. Thus, the introduction of self-healing data pipeline architecture is expected to eliminate those limitations through continuous monitoring of system performance, detection of anomalies, and diagnostics and correction of detected errors.

In turn, self-healing data pipelines are shown to effectively improve the accuracy of anomalies detection and decrease the recovery time while increasing the level of system availability as a result of implementing advanced techniques. For example, such methods include machine learning-powered detection of deviations, checkpointing and restoration, automatic retries, dynamic workflow routing, and adaptation of computing resources allocation. Consequently, self-healing pipelines enable maintaining high-quality data, ensuring business continuity, and performing massive analyses or machine learning tasks more efficiently.

It is recommended to consider further research on developing AI-based root-cause analysis solutions that would allow detecting the reasons for errors automatically and explaining decisions made by artificial intelligence. Another direction could be exploring reinforcement learning-based approaches to recover data pipeline operation. In this approach, agents learn to identify the most efficient ways to recover the system operation based on its history and changing environment. Finally, development of Autonomous DataOps platforms that combine multiple functions such as monitoring, anomaly detection, recovery, governance, and optimization is another interesting area to explore.

References

- Kephart, J. O., & Chess, D. M. (2003). The vision of autonomic computing. *Computer, 36*(1), 41–50.
- Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys, 41*(3), 1–58.
- Gupta, M., Gao, J., Aggarwal, C. C., & Han, J. (2014). Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering, 26*(9), 2250–2267.
- Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys, 46*(4), 1–37.
- Krishnan, S., Wang, J., Wu, E., Franklin, M. J., & Goldberg, K. (2016). ActiveClean: Interactive data cleaning for statistical modeling. *Proceedings of the VLDB Endowment, 9*(12), 948–959.
- Rekatsinas, T., Chu, X., Ilyas, I. F., & Ré, C. (2017). HoloClean: Holistic data repairs with probabilistic inference. *Proceedings of the VLDB Endowment, 10*(11), 1190–1201.
- Breck, E., Cai, S., Nielsen, E., Salib, M., & Sculley, D. (2017). The ML test score: A rubric for ML production readiness and technical debt reduction. *IEEE Big Data*.
- Schelter, S., Lange, D., Schmidt, P., Celikel, M., Biessmann, F., & Grafberger, A. (2018). Automating large-scale data quality verification. *Proceedings of the VLDB Endowment, 11*(12), 1781–1794.
- Breck, E., Polyzotis, N., Roy, S., Whang, S. E., & Zinkevich, M. (2019). Data validation for machine learning. *Proceedings of Machine Learning and Systems*.
- Schelter, S., Grafberger, A., Dunning, T., & others. (2019). Unit testing data with Deequ. *Proceedings of the ACM SIGMOD International Conference on Management of Data*.
- Breck, E., Zinkevich, M., Polyzotis, N., Whang, S. E., & Roy, S. (2020). Data analysis and validation in continuous ML pipelines. *Proceedings of the ACM SIGMOD International Conference on Management of Data*.
- Psaier, H., & Dustdar, S. (2011). A survey on self-healing systems: Approaches and systems. *Computing, 91*, 43–73.