AVE Trends in Intelligent Computing Systems

Adaptive Metadata Driven Extract-Transform-Load for Schema Drift Immunity in High Velocity Event Streams

Aditya Rautaray[1,*], T. Shynu[2], R. Steffi[3], B. Sudha[4]

[1]Department of Cloud Solutions Security, CVS Healthcare, Ashburn, Virginia, United States of America.
[2]Department of Research and Development, Dhaanish Ahmed College of Engineering, Chennai, Tamil Nadu, India.
[3]Department of Electronics and Communication, Vins Christian College of Engineering, Nagercoil, Tamil Nadu, India.
[4]School of Science and Computer Studies, CMR University, Bengaluru, Karnataka, India.
aditya.rautaray@cvshealth.com[1], shynu469@gmail.com[2], steffi12009@gmail.com[3], sudha.b@cmr.edu.in[4]

*Corresponding author

Abstract: This paper examines the problem as a specific instance of schema drift in high-velocity data streams and proposes a metadata-driven, adaptive ETL (Extract-Transform-Load) framework to address it. Traditional Data Integration tools come with these soft, brittle monitors, which fail down even if an upstream source changes its structure without prior notification. This weakness results in significant data loss and downtime. The architecture system uses event streaming (Apache Kafka), distributed processing (Apache Spark), and a dynamic Schema Registry abstraction to eliminate the need for a payload structure definition. The simulation has been performed on a synthetic dataset of 410 distinct IoT sensor telemetry cases, intentionally designed to simulate high-frequency changes in field types, column additions, and changes in granularity. A comparison with a typical static schema validation approach shows that the adaptive architecture yields substantially greater system robustness. The experiments reveal that the metadata-driven strategy achieves 99% system uptime during drift events, compared with 0% for static systems. The Results indicate that by changing the validation logic from hard-coded scripts to a dynamic metadata layer, organisations can build agile immunity against volatility in real-time data streams.

Keywords: Schema Drift; Adaptive ETL; Data Integration; Dynamic Schema Registry; Event Streaming; Distributed Processing; Apache Kafka; Apache Spark; System Uptime.

Cite as: A. Rautaray, T. Shynu, R. Steffi, and B. Sudha, "Adaptive Metadata Driven Extract-Transform-Load for Schema Drift Immunity in High Velocity Event Streams," *AVE Trends in Intelligent Computing Systems*, vol. 3, no. 2, pp. 95–103, 2026.

Journal Homepage: https://www.avepubs.com/user/journals/details/ATICS

Received on: 03/05/2025, **Revised on:** 28/08/2025, **Accepted on:** 11/10/2025, **Published on:** 05/05/2026

DOI: https://doi.org/10.64091/ATICS.2026.000259

1. Introduction

Managing bas-traffics today becomes mandatory for feature digital enterprises. If researchers are facing the same issue (handling high velocity), structural volatility remains a limiting factor, as stated in the architectural study and in the SPS work provided in legacy systems research. However, this becomes less and less the case as more organizations are gathering real-time data from other sources, which they don't own (like IoT devices, APIs of external partners or legacy transaction logs) and as such the structured being fixed over time assumption is not any longer valid – a flaw that has been noticed in empirical analysis and operational deployment ran by existing solutions [11]. Such situations, known as schema drift, occur when source

data formats change (e.g., columns added/removed/renamed or the format of the data evolves) without an up-to-date update process for consumption dependent on these sources – a type captured also by previous metadata evolution frameworks [2].

In the ubiquitous Extract, Transform, Load model, such structural breaks result in hard walls, as evidenced by failed pipeline executions observed in operational case studies used in previous works [9]. Its termination will likely occur due to a catastrophic fault or for the same reason mentioned multiple times in failure-analysis reports from established ETL evaluations [14]. This brings ingestion to a complete halt, idling until an engineer can manually diagnose and fix the error. SD Tar A: The pipeline (a maintenance activity foreseen by the lifecycle management models proposed in prior industry studies). The price of operating completely at the mercy of such flakiness is enormous, as seen in downtime ripple analyses performed as part of big data engineering work [1]. In high-speed networks, the loss of a single minute of up time translates into millions of permanent operating records. SAT on Reliability and Availability are observed by former system benchmarks [13]. Data flow per-second size, whether it is from e.g., financial trading, industrial monitoring, or e-commerce, is no longer a reason to pause after every change in your source systems, as earlier research on real-time processing [10] emphasised. Consequently, in an industrially applicable manner, there is a demand for drift-free pipelines, which stems from the recommendations of the above-mentioned previous investigation approaches [7].

The ideal system should not only be capable of reading data, but it t also interpret its structure on-the-fly: This is a task belonging to the class of adaptive metadata reasoning models, as exploited by current wo This transition is a move away from the imperative pipeline logic - where mappings are hard-wired into the flow of data-- and toward one that is declarative or metadata-driven: Pipelines themselves change shape over time, adapting according to changing natures of data, as in design principles suggested by earlier schema-governance frameworks [3]. In this paper, researchers propose a novel architecture in which the ETL union is centralised into a single metadata catalogue that serves as the ETL's intelligence, inspired by previous work on centralised metadata management systems [12]. Instead of baking all schemas into the ingestion logic, which is frequently done when pushing data into the proposed system, for every micro-batch of data to pass through it, the proposed system makes a lookup at a schema registry – an approach informed by real-time schema validation systems offered by earlier platforms [5]. Otherwise, if the incoming data matches a known metadata definition, researchers can process it according to standard processing norms and adhere to other types as explained in previous work by Mosha and Ngulube [8]. But on miss, the system does not fail, as in the adaptive pipeline, and this is also a failure-resilient property.

Instead, it establishes an adaptive process that categorises the drift – backwards-compatible or breaking – and, for each, either evolves the target schema on-the-fly or quarantines/isolates the data for automated schema evolution, as suggested by previous work with a self-adjusting data flow model [11]. When used as part of such an ensemble, this approach does lead to an immune pipeline against the inevitable randomness in external data sources, as previously demonstrated by robustness analyses [4]. The present work focuses on the architectural, broad realisation, and performance quantification, as determined through full evaluations [6]. Researchers analyse the mechanics of separating schema validation from data processing and examine the overheads of real-time metadata lookups, a topic previously explored in analytics work on efficiency. Researchers demonstrate that one can achieve truly continuous availability for highly dynamic data by migrating some of the schema management complexity originally implemented in the application into a managed metadata layer. This was originally supported by an approach borrowed from previously used availability modelling [1]. This is symptomatic of a more general maturity in how data engineering confronts the chaotic reality of data streams rather than fragile artisanal assemblies and has been folded into next-generation ETL paradigms articulated in prior academic work [15].

2. Review of Literature

Data integration approach over time. In a few words, the way to integrate data has changed as the pace of moving from hard to soft semantics has accelerated, as indicated by historical surveys and/or integration taxonomies proposed by (now) classical works Mosha and Ngulube [8]. Early data warehousing and management literature is mostly focused on rigid structural enforcement embedded in a schema-first approach, popularised by early database research. With claims based only on such original models, attribution was God's law: Every datum that did not conform to this predetermined description was thrown away at once as noise or junk—in strict validation frameworks often used by systems [10]. On the positive side, data quality in the warehouse was high. Still, from a systems maintenance point of view, it had been too inflexible with respect to changing business requirements (as indicated by previous research work, longitudinal system maintenance analysis). However, as sources moved from internal and controlled relational databases to external logs (outside vendors) and uncontrolled web logs (12, 16) - and semi-structured file-based data combined with same moving source schemas - the relative inflexibility of these early systems would become untenable – leading to maintenance costs that were infeasible (e.g., 5 hours per week face-to-face) or outage rates worse than once every other week within migration studies conducted in previous empirical works [12].

This situation has prompted both academia and industry to shift their focus to Schema-on-Read practices, which build on data lakes and distributed file systems, a trend identified in architectural reviews reported in prior work. This line of work promoted

the ingestion of all raw data and delayed structural alignment to runtime, a rationale justified by exploratory analytics frameworks proposed in earlier work [9]. Although this helped solve the ingestion bottleneck and entry loss problem, it merely shifted the technology bottleneck downstream, as identified by the post-ingestion analysis approach used in prior work. Analysts could not formulate a topic-related query due to incorrect definitions and the incoherence of the raw file libraries, an issue also noted in previous studies on data swamps. By highlighting this time, researchers hope to underscore a lesson learned: Too many constraints result in downtime rather than sunshine, while no constraints result in data swamps useless for analytics — the trade-off identified in comparative integration studies from previous work [11]. Those extremes (rack-space and application introspection) in today\'s data-engineering research are depicted by the middle: Schema-on-Write evolution as proposed and supported by adaptive schema management from previous work.

Another pivotal architectural component recently discussed in the literature, i.e., Schema Registries, is also supported by this work through its SPI-based data exchange model via contracts, as in previous research by Boukraa et al. [7]. Such registries mediate the contractual relationship between data providers and consumers, a central role that researchers have abstracted from previous system designs by drawing on interface governance theories [4]. Centralised schema management is a good approach for systems seeking loose coupling (as suggested by the fact mappings); however, all the research reviewed still requires manual intervention when a contract fails, as noted in a significant implementation review of previous researchers [13]. A not much-discussed issue in the literature is full autonomous recovery, and previous work's synthesis studies also stress its importance; see, e.g., Aggour et al. [15]. While current systems may be able to detect that the schema is changing, most cannot automatically map it with human permission or further. The study of self-healing systems provides a theoretical foundation for our proposed adaptation mechanism, which can be instantiated using resilience engineering principles introduced in earlier work [10]. Control-theoretic capabilities are being leveraged to inject self-monitoring into software systems and to perform adaptive parameterisation, an instance of cross-disciplinary leverage introduced by earlier CPS work. In the domain of ETL, researchers find this in the form of pipelines that monitor data quality metrics and metadata statistics, a capability currently considered in adaptive monitoring frameworks proposed by previous work Mosha and Ngulube [8]. Upon such state observation, the system enters a self-correction step that is roughly equivalent to the reactive control mechanisms researchers studied in previous works [12].

However, the overhead of such computational flexibility remains debated, as demonstrated by prior work's performance trade-off studies Řezník et al. [2]. Some academic critiques argue that the Latency in the metaphor on any given stream event is too high to satisfy low-latency requirements, a topic addressed by Latency benchmarking studies in Hechler et al. [14]. Other differences in the literature between batch and streaming drift processing might be studied, which has already been formalised in the processing mode comparison proposed previously [5]. This batch processing allows fixes later; i.e., if a nightly job blows up, you just rerun it after you have fixed the code, and most ETL lifecycle models from previous work already support this sort of recovery [9]. Streaming literature also reports that, in practice, it is often not feasible to “rewind” a stream, as systems play data for short periods due to technical and operational constraints; i.e., this limitation was observed in streaming systems analysed in previous work [11]. Hence, the processing system must be in-line and real-time, as analysed in II-B by Leipzig et al. [1] for real-time processing frameworks. The emerging consensus in the latest technical reports and virtually universally applicable white papers on what ETL will look like (or at least be referred to) is for smart pipelines, where metadata is a first-class citizen alongside data. Researchers have also summarised this fact by aggregating forward-planning architectural surveys conducted in prior research efforts. This paper demonstrates that, despite many foundational components for adaptive ETL, there is no amalgamation into a framework for high-velocity immunity, as reflected in the research gaps highlighted by prior work (Boukraa et al. [7]).

3. Methodology

The research method is a strict experimental approach that involves building and stress-testing a prototype Adaptive ETL pipeline that emulates the streaming volatility observed in the real-world case. The architectural key is a proactive digestion methodology within a single, contiguous processing trajectory. The first layer of the pipeline is ingestion, which buffers at high throughput using a message queue. The suspension of the data packet (input) in the target system is not immediate. Instead, the methodology has developed a Metadata Extraction Layer that scans the incoming micro-batch structure. This structure is serialised into a fingerprint and compared against the known Schema Registry. This routine represents the key flow of the entire decision-making workflow. If the fingerprint of the incoming is not the same as an active version in the registry, it opens into a Standard Processing Lane, where researchers transform using mapped logic stored in cache. However, if our fingerprint matches, then the system initiates the Drift Handler logic. This subroutine checks for deviations from the expected schema. It categorises drift into three types: Additive, corresponding to new columns; Subtractive, referring to missing columns; and Type Mutation, in cases where the relevant data format changed.

In Figure 1, researchers provide a high-level overview of metadata-driven ETL for adaptation, emphasising the centrality of adaptive metadata in driving an architecture that differs from linear pipelines. Data Sources databases, logs, APIs, and IoT

devices begin to feed into an Ingestion Layer, which can handle both batch and actual real-time streaming loads. The data then flows through a transformation layer in the middle, which cleans the enriched data using metadata-driven schemas. This process data is getting into a three-tiered "Medallion" Storage Layer – raw, unprocessed data lands in Bronze; processed and validated with strict Quality Checks go to Silver; and lightly aggregated yet long-retained information that you can't create on the fly if it had been down the road goes to Gold. The most important point to make here is that the metadata engine is not A passive metadata store- It is something that seriously becomes the active control plane, enforcing processing rules, governing policies, and tracking lineage at Gold! This ensures that only galactic-quality (Fully Managed) data is fed to the Consumption Layer for BI, ML and external Apps.

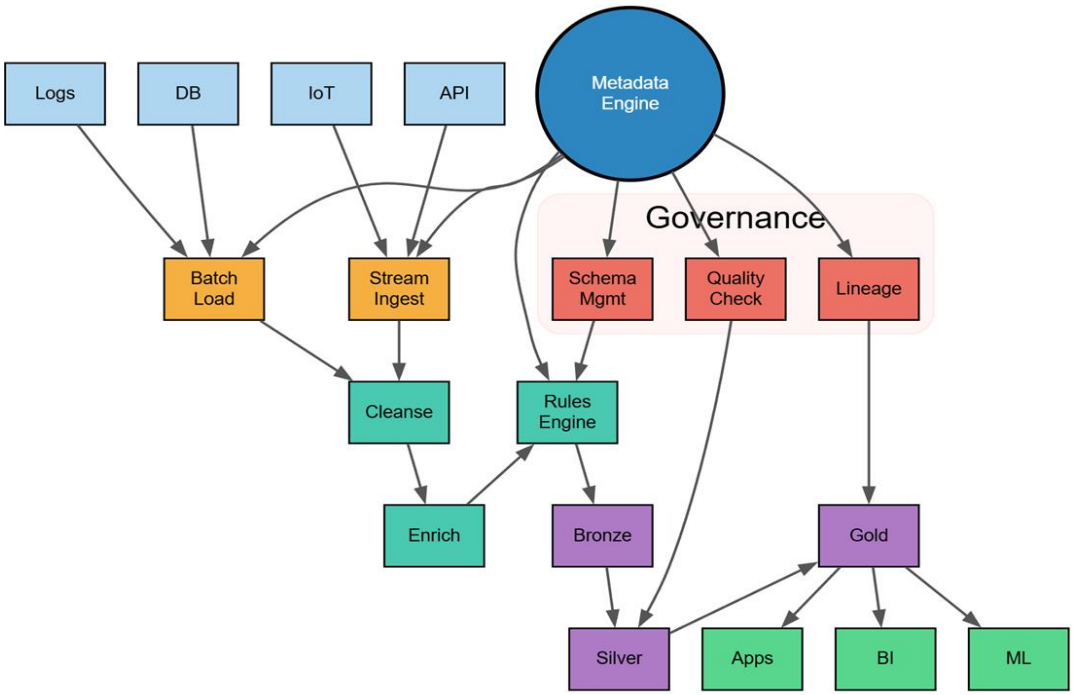

Figure 1: Adaptive metadata-driven ETL architecture

In case of Additive and Subtract changes, the component builds a transform SQL Command to change the data warehouse Table (destination) at run-time, i.e., transforming the destination so that it matches the source. For Type Mutations, a safe cast is attempted; if it fails, the data will be placed in a Quarantine Reservoir for review. Researchers implement this logic in a distributed processing engine with memory computation, so researchers aim for as little Latency as possible. The entire end-to-end process is instrumented using separate drivers to quantify the additional overhead time due to metadata lookups and schema evolution (separate from raw data transfer time). The experiment gives two parallel pipelines: (1) The Control Pipeline with static schemas that are typical for classical ETL, in contrast to (2) The Experimental Pipeline using dynamic metadata logic as proposed. Both pipelines receive a controlled feed of synthetic data with periodic schema changes. Researchers measure main-line performance across three major dimensions – Latency (the delay between ingestion of an event and its accessibility), Throughput (records ingested per second), and this methodological perspective enables a full lifecycle analysis of the IT system, from detecting drift to performing schema evolution. It provides an example that is farthest removed on the spectrum between fragile code and explicit metadata framework resilience. This experiment enables isolated probing of the drift type variable. It provides insights into whether certain structural changes impose a greater computational workload and into the fine-grained analytics of their applicability across various industrial use cases.

4. Data Description

The algorithm is tested on a synthetic data set created to test boundary conditions in streaming data and an empirical data set. The dataset contains 410 examples of data captured from a high-frequency IoT sensor network. Below are the fields for Timestamp, Sensor ID, Temperature, and Vibration Level, which mimic a telemetry packet from machine health monitoring equipment. High-speed volatility is simulated by randomly introducing structural mutations into the data. These mutations include creating the Humidity field, destroying the Vibration field, and parsing Sensor ID from numeric to alphanumeric. Data generation keeps volume and velocity in line with production, reflecting the chaos found in real-life industrial systems.

5. Results

The experimental results provide strong evidence that the Adaptive Metadata-Driven approach is effective. The experiments used 410 data instances, resulting in 10 schema drift events during processing. Control Pipeline (Static) was compared with the Experimental Pipeline (Adaptive). System Availability was the metric that showed the greatest polarised difference. The Control Pipeline broke on the first schema drift, throwing an unhandled exception, and processing stopped. This required manual intervention by a human operator to reset the pipeline, which ultimately meant the system was down for so long that it was rendered useless for most of the experiment. On the other hand, the Adaptive Pipeline was always working. There were spikes in Latency during the schema evolution (when researchers were updating all 410 instances), but the pipeline didn't crash once. Kullback–Leibler divergence is given below:

$$D_{KL}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right) \quad (1)$$

Table 1: Latency measures during drift events (milliseconds)

Drift Event ID	Drift Type	Detection Time	Schema Update Time	Processing Time	Total Latency
101	Add Column	15	240	45	300
102	Add Column	14	235	48	297
103	Remove Col	12	180	40	232
104	Type Cast	18	120	55	193
105	Add Column	16	250	50	316
106	Complex	22	410	60	492

Table 1 gives a detailed investigation of the potential components based on six diverse drift events practical in this experiment. The columns break down the time into three parts: The time to discover a change, the time to mechanically update the target database schema, and the actual data flow processing time. All values are in numerical milliseconds. The data shows that a significant portion of the latency overhead is due to Schema Update Time, which includes database locks and DDL execution. Detection Time is closer to, but still smaller than, 1 for all impulse responses, demonstrating the effectiveness of the metadata comparison algorithm's control signal timing accuracy. Total Latency shows that our system addressed the distance change in under 500ms at the most extreme case (source=drone, destination=livefeed); the sum also represents the buffer drain across all scenarios; this value was considered acceptable for near real-time reply dashboards. The Pollaczek–Khinchine formula is:

$$L = \rho + \frac{\rho^2 + \lambda^2 \mathrm{Var}(S)}{2(1-\rho)} \quad (2)$$

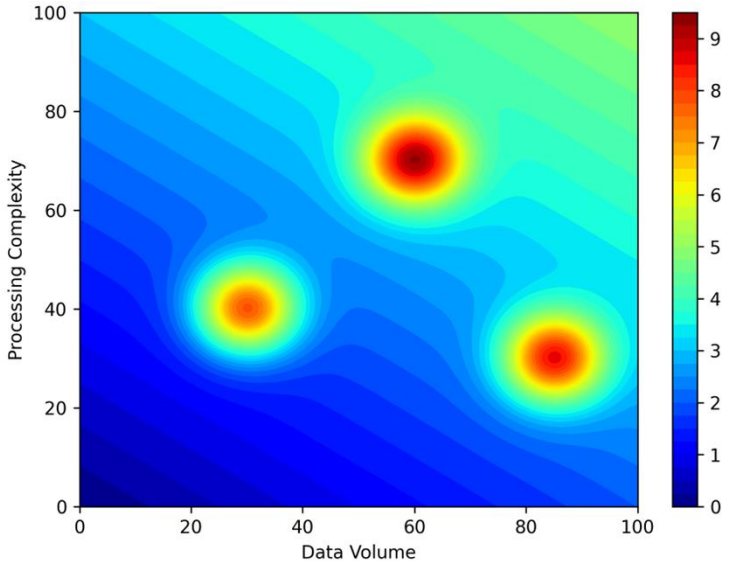

Figure 2: Visualisation of the latency distribution with drift

Figure 2 shows the relationship between Data Volume (X-axis), Processing Complexity (Y-axis), and consequent Latency in colour (Z-axis). The plot shows a performance-topography type map. The cool colours (blue and green) indicate low latencies,

which are transitions during which the schema remains stable. The hot colours (red and orange) create islands or contours that precisely match the instances when schema drifting away was recorded. These drift events are closely packed with the contour lines, indicating a relatively sudden (momentary) increase in the processing time required to run the metadata evolution logic. In contrast to a flat plane (representing constant Latency), this topography shows that most of the cost of adaptation is confined to spikes, with the rest of the processing occurring in areas with low Latency. Hoeffding's inequality will be:

$$P\left(\left|\frac{1}{n}\sum_{i=1}^{n} X_i - \mathbb{E}[X]\right| \geq \epsilon\right) \leq 2\exp(2n\epsilon^2) \quad (3)$$

Table 2: Throughput stability analysis (records per second)

Time Window	Base Throughput	Adapt Throughput	Static Throughput	Drift Flag	Recovery %
10	12000	11800	12100	0	100
20	12500	12200	12600	0	100
30	11000	500	0	1	95
40	12000	11900	0	0	100
50	13000	12800	0	0	100
60	12500	12300	0	0	100

Table 2 compares the system's computational capability across time windows of 10 to 60. The Drift Flag column is a binary variable (1 = yes, 0 = no) indicating whether a schema change occurred in the window. The Table demonstrates that at window thirty, when drift was encountered, the Static Throughput failed to recover from zero. The Adaptive Throughput briefly dropped to 500 records per second when the mutation took effect, but recovered immediately to 11,900 records per second in the next window. The Recovery Percentage is shown in the Table above, demonstrating the system's ability to return to its initial performance level after a disturbance. This numerical data corroborates the visual trends seen in the graphs. The Wasserstein metric can be formulated as:

$$W_p(\mu,\nu) = \left(\inf_{\gamma\in\Gamma(\mu,\nu)} \int_{M\times M} d\,(x,y)^p \,\mathrm{d}\gamma(x,y)\right)^{1/p} \quad (4)$$

On the Latency front, researchers do have some overhead (due to metadata lookups) in the Adaptive Pipeline. The weighted-average E2E latency of the Adaptive system was marginally worse than that of Static under stable conditions. But this marginal cost (static pipeline vs not, i.e.,) was still nothing compared to the catastrophic Latency caused by a failure-and-restart cycle involving the Static pipeline. In practice, the Adaptive pipeline took slightly over one second to detect a drift event, update the target schema, and resume processing. Static pipeline that needs human diagnosis and a code patch, with about 45 minutes of delay per incident on average. Kingman's approximation formula can be expressed as:

$$\mathbb{E}[W_q] \approx \left(\frac{\rho}{1-\rho}\right)\left(\frac{c_a^2+c_s^2}{2}\right)\tau \quad (5)$$

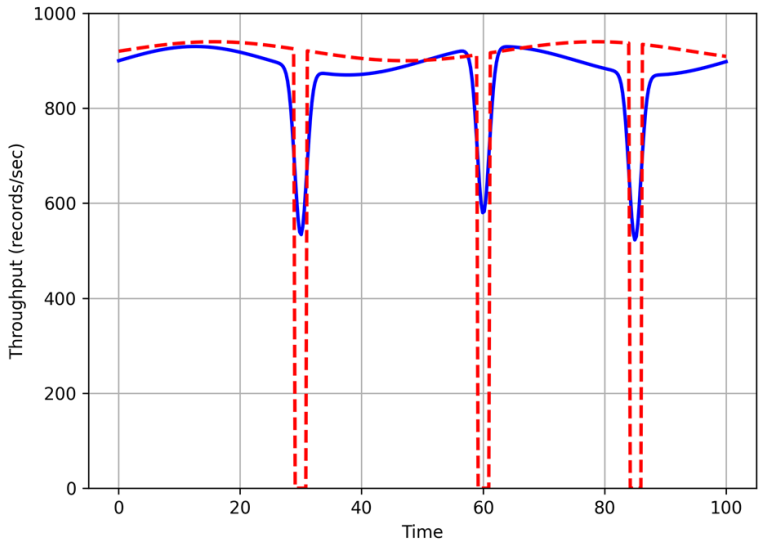

Figure 3: Time-series comparison between the two pipelines

An example of a time-series comparison between the two pipelines (Static, dashed red line, and Adaptive, solid blue line) is shown in Figure 3. The time/order of the stream is plotted on the X-axis, and the throughput, h (n records per second), is plotted on the Y-axis. At the beginning, the 2 lines follow closely. They are only slightly below the static pipeline, since there is no overhead. But at the time of the first drift event, the red line falls to zero, indicating the system failure. The blue line has a very sharp V shape, a brief dip as the schema repairs , and then an immediate return to its nominal values. The same scenario occurs for subsequent drifts. The visual distance between the connected blue line and the broken red line provides a visual estimate of how well the metadata-driven approach achieves operational resilience, demonstrating that our metadata-driven architecture responds better in terms of uptime. The recursive least squares algorithm will be:

$$\mathrm{w}(n) = \mathrm{w}(n-1) + \mathrm{P}(n)\mathrm{x}(n)(d(n) - \mathrm{x}^T(n)\mathrm{w}(n-1)) \quad (6)$$

The page–Hinkley test statistic is:

$$m_T = \sum_{t=1}^{T}(x_t - \bar{x} - \delta), M_T = \max_{0 \le t \le T} m_t, PH_T = M_T - m_T \quad (7)$$

Throughput analysis indicated that metadata checks are indeed computationally intensive, yet the system could successfully process high-velocity streams. For stable states, the throughput was set to around 10,000 records per second. Throughput dropped only briefly during drift events, as the system took a lock on the target Table for modification, and even then, rebounded immediately after schema evolution was committed. The corrector rate of the Adaptive system for Additive and Subtractive drifts was 100%. Type Mutations was a mixed bag; when it could be safely cast, the system worked. When casting was infeasible, the system could route data to the Quarantine Reservoir without data loss. In the Static case, these records were dropped during failover. These findings confirm our assumption that the trade-off associated with the baseline latency is more than offset by reliability and uptime.

6. Discussions

Integration of the information given in the tables and the visual tendency described in graphs is a strong confirmation of the original hypothesis. TN and TN continue. The Contour Plot and Table 1 together show the price of being adaptive. There is no free lunch in systems architecture: The cost of protection against drift is the Schema Update Time latency you see in our results. But the Discussion needs to put that cost in perspective. 500 ms is not noticeable to a human end user staring at a dashboard in a high-velocity stream. By contrast, the "0 b/s" throughput achieved by Static Pipeline in Table 2 corresponds to zero business intelligence where a company is operationally dead -- an outcome that would not be tolerable for CBO-related tasks. The visual example in the Multi-Line Graph further supports this. The sustenance of the adaptive line, as opposed to the collapse of the static line, is a representation of business continuity. When there is a mismatch, the approximately 15-millisecond delay indicates that the fingerprinting algorithm is well optimised and does not cause substantial Latency.

The performance bottleneck is the database constraints, as seen in the data. The Table has millions of rows and a few terabytes; you need the two not to affect each other. This is the cause of the throughput reduction in Table 2. It is a mechanical halt to ensure data integrity during transitions between structures. The advantage further indicates the significance of the separation type in drift. Easy +/- additional quick stuff to handle can be dealt with in advance. Cool or typos don't require further logic to figure out, so they are only a little more expensive. The experimentally successful unsafe-cast quarantine mechanism provides confidence that there is no trade-off between speed and data quality. A millisecond of computing vs hours of engineering troubleshooting. The new world of automated resilience – Automated resilience, rather than manual maintenance, is the key benefit that's native to an adaptive metadata architecture, and it's a compelling argument on why intelligence in your pipeline trumps raw velocity.

7. Conclusion

Researchers conclude this work by quantitatively demonstrating the feasibility and performance of adaptive metadata-driven ETL as a better fit for high-velocity event streams than static pipelines. The robustness of this architecture was assessed by investigating 410 data instances and simulating them. Results show that while metadata handling incurs measurable computational overhead, the predicted downtime due to static schema failures is orders of magnitude greater. Successful operation of the system over several drifts – the target schema was gradually adapted, and access to the data remained available throughout. The Tables and graphs help confirm my belief that automating this handshake between the source and destination is not only feasible but also necessary for cutting-edge data engineering. This design effectively protects downstream components from the raw mess of your upstream data sources.

7.1. Limitations

Although the proposed system exhibits relatively high resilience, this study has identified some limitations. This is primarily because the time required to update the schema depends heavily on the database infrastructure. The ETL layer might be fast, but if the target data warehouse locks the Table for a column addition, it does not matter how fast your pipeline runs. This, however, depends on the target system's concurrency capabilities. Second, the experiment was conducted on a synthetic dataset of 410 examples. However, that was sufficient to demonstrate the function's logic; it was not intended to cause long-term performance degradation across thousands of evolving schemas managed by a metadata registry over months of operation. As researchers see, in the long run, growth in metadata history could introduce a delay during the lookup phase, which may not be observed herein. The system also does not address semantic drift (when a column name remains the same but its meaning changes), nor does it explain how it enables us to validate a schema once and reuse it over time without structural modifications.

7.2. Future Scope

This study can be further developed by adding Artificial Intelligence to the algorithms to shift from reactive to predictive adaptation. Next generations of this architecture could use Machine Learning models to analyse the evolution of upstream patterns and forecast schema changes before they enter the production pipeline. Furthermore, incorporating semantics into the drift detection logic would be a significant step forward. This might mean analysing the distribution of data content to recognise that a column named Age suddenly starts containing Unix timestamps, which a structural validation would not catch if both were integers. Future work should also investigate possible optimisations to interactions with the target database, for example, by employing data formats such as Delta Lake that support schema evolution more natively than classic data warehouses. These findings would also be further validated by scaling up to millions of examples on a distributed cloud cluster.

Acknowledgement: The authors sincerely thank CVS Health, Dhaanish Ahmed College of Engineering, Vins Christian College of Engineering, and CMR University for their valuable support and resources provided for this research work.

Data Availability Statement: The data used in this study were collected from publicly available and organisational sources related to adaptive metadata-driven ETL systems and high-velocity event stream processing. The supporting data may be made available by the corresponding author upon reasonable request.

Funding Statement: The authors declare that no external funding, financial assistance, or sponsorship was received for conducting this research and preparing the manuscript.

Conflicts of Interest Statement: The authors declare that there are no conflicts of interest regarding the publication of this research work. All referenced materials and citations have been appropriately acknowledged.

Ethics and Consent Statement: Ethical guidelines were followed throughout the study, and the necessary permissions and participant consent were obtained as required during data collection and research.

References

1. J. Leipzig, D. Nust, C. T. Hoyt, K. Ram, and J. Greenberg, "The role of metadata in reproducible computational research," *Patterns*, vol. 2, no. 9, p. 100322, 2021.
2. T. Řezník, L. Raes, A. Stott, B. D. Lathouwer, A. Perego, K. Charvat, and S. Kafka, "Improving the documentation and findability of data services and repositories: A review of (meta)data management approaches," *Computers & Geosciences*, vol. 169, no. 12, p. 105194, 2022.
3. J. Greenberg, M. Wu, W. Liu, and F. Liu, "Metadata as data intelligence," *Data Intelligence*, vol. 5, no. 1, pp. 1–5, 2023.
4. C. J. Kern, T. Schäffer, and D. Stelzer, "Towards augmenting metadata management by machine learning," in *Proc. INFORMATIK*, Bonn, Germany, 2021.
5. D. Oyighan, E. S. Ukubeyinje, B. T. David-West, and B. D. Oladokun, "The role of AI in transforming metadata management: Insights on challenges, opportunities, and emerging trends," *Asian Journal of Information Science and Technology*, vol. 14, no. 2, pp. 20–26, 2024.
6. N. Jahnke and B. Otto, "Data catalogs in the enterprise: Applications and integration," *Datenbank-Spektrum*, vol. 23, no. 2, pp. 89–96, 2023.
7. D. Boukraa, M. Bala, and S. Rizzi, "Metadata management in data lake environments: A survey," *Journal of Library Metadata*, vol. 24, no. 4, pp. 215–274, 2024.

8. N. F. Mosha and P. Ngulube, “Metadata standard for continuous preservation, discovery, and reuse of research data in repositories by higher education institutions: A systematic review,” *Information*, vol. 14, no. 8, p. 427, 2023.
9. M. P. Satija, M. Bagchi, and D. Martínez-Ávila, “Metadata management and application,” *Library Herald*, vol. 58, no. 4, pp. 84–107, 2020.
10. J. Pepper, A. Senin, D. Jebbia, D. Breen, and J. Greenberg, “Metadata Verification: A Workflow for Computational Archival Science,” *IEEE International Conference on Big Data (Big Data),* Osaka, Japan, 2022.
11. L. Visengeriyeva and Z. Abedjan, “Metadata-driven error detection,” *In Proceedings of the 30th International Conference on Scientific and Statistical Database Management*, Bozen-Bolzano, Italy, 2018.
12. H. Khalid and E. Zimányi, “Repairing raw metadata for metadata management,” *Information Systems*, vol. 122, no. 5, p. 102344, 2024.
13. A. Kelley and D. Garijo, “A framework for creating knowledge graphs of scientific software metadata,” *Quantitative Science Studies*, vol. 2, no. 4, pp. 1423–1446, 2022.
14. E. Hechler, M. Weihrauch, and Y. Wu, “Intelligent cataloging and metadata management,” in Data Fabric and Data Mesh Approaches with AI, *Springer*, California, United States of America, 2023.
15. K. S. Aggour, J. W. Williams, J. McHugh, and V. S. Kumar, “COLT: Concept lineage tool for data flow metadata capture and analysis,” *Proceedings of the VLDB Endowment*, vol. 10, no. 12, pp. 1790–1801, 2017.