Journal of Advances in Developmental Research (IJAIDR)

E-ISSN: 0976-4844 ● Website: www.ijaidr.com ● Email: editor@ijaidr.com

Challenges in Developing a Real-Time Root Cause Analysis Platform Using Big Data and Machine Learning

Sujay Kanungo

Independent Researcher
Boston, USA
sujay2002@gmail.com

Abstract:
The rapid evolution of technology has led to an exponential increase in data generation, making real-time root cause analysis (RCA) a critical necessity for organizations striving for operational excellence. This paper explores the multifaceted challenges encountered in the development of a real-time RCA platform utilizing big data and machine learning techniques. Key obstacles include data integration from diverse sources, ensuring data quality and consistency, and the complexity of algorithm selection for accurate anomaly detection and diagnosis. Furthermore, the paper discusses the limitations of existing machine learning models regarding scalability and adaptability in dynamic environments. The interplay between real-time data processing requirements and the computational demands of advanced analytics is also examined. By identifying and analyzing these challenges, this study aims to provide insights and potential strategies for overcoming the barriers to effective real-time RCA implementation, ultimately contributing to more resilient and responsive organizational frameworks.

Keywords: Root Cause Analysis, Machine Learning, Real time RCA.

I. INTRODUCTION

The rapid proliferation of interconnected devices, particularly within the Internet of Things (IoT), has catalyzed the development of innovative applications across diverse domains, including smart homes, Industry 4.0 environments, and autonomous vehicles. These contexts are increasingly characterized by complex and often system-of-systems architectures composed of heterogeneous components. As these systems evolve, they generate data at unprecedented rates. When left unmonitored or insufficiently managed, this data surge can precipitate real-time, data-centric systemic failures. Consequently, there is a growing imperative to proactively mitigate major incidents and to detect, diagnose, and address minor anomalies before they escalate into service-affecting failures.

This shift reflects a broader transformation in analytical value propositions, moving from traditional postmortem Root Cause Analysis (RCA)—which seeks to understand anomalies after they occur—toward *preventive* and *real-time* diagnostic approaches. The emerging paradigm focuses on identifying early causal precursors of failure, thereby inhibiting the progression of small anomalies into significant operational disruptions. To articulate this evolution, we refer to the overarching concept as **Real-Time Preventive Root Cause Analysis (RTP-RCA)**, a forward-looking, causality-driven analytical framework designed to function continuously within modern connected environments.

Although real-time RCA remains an ambitious objective and is currently explored primarily in academic research, a viable architectural foundation appears attainable, particularly when integrated with the analytics principles of pervasive and continuous analysis. RTP-RCA is envisioned to operate across the full landscape of user- and system-generated events, responding dynamically to analytical demands with high

prioritization in an era defined by ubiquitous connectivity. This paradigm synthesizes preventive RCA with advanced temporal and causal modeling to support real-time diagnostic reasoning.

Several data-driven methodologies contribute to the RTP-RCA framework, including Statistical Process Control and time-series analysis, artificial intelligence and deep learning–based event correlation, graph-theoretic propagation detection, forensic approaches such as inverse machine learning, and emerging small-sample deep learning strategies. Each of these methodologies represents a major analytical pathway within the broader scope of modern RCA and is integral to the preventive, forward-focused orientation of RTP-RCA.

As contemporary interpretations of "root cause" broaden—parallel to the dilution of strict boundaries around system failures and anomalies—the objectives of RCA have similarly expanded. RTP-RCA therefore encompasses not only the identification of direct causal agents but also the discovery of auxiliary scenarios that may escalate into user-visible disruptions. In doing so, it extends the reach of traditional RCA into a comprehensive, anticipatory, and system-wide analytical capability for modern interconnected environments.

Despite its promise, the development of an effective Real-Time Preventive Root Cause Analysis (RTP-RCA) system presents several substantive research and engineering challenges. Chief among these is the need to manage the scale, heterogeneity, and velocity of data produced by complex cyber-physical and socio-technical systems. Real-time causal inference requires computationally efficient algorithms capable of handling high-dimensional, multimodal data under stringent latency constraints. Moreover, ensuring the reliability, interpretability, and robustness of causal models in dynamic and partially observable environments remains a nontrivial task. Additional challenges arise from the integration of diverse analytical methods—spanning statistical modeling, machine learning, graph analytics, and domain-specific heuristics—into a cohesive framework that can operate autonomously and adaptively. The study therefore aims to systematically identify and characterize these challenges, with the goal of informing architectural guidelines, methodological innovations, and evaluation criteria that support the realization of practical RTP-RCA systems.

II. ARCHITECTURAL CONSIDERATION FOR RTP-RCA

Architecture refers to the fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution. A framework that at some level can only loosely be described as big-data streaming-analytics architecture is being built in collaboration with the external technology partner. Real-time or fast root-cause analysis requires eliciting the semantically most relevant information from critical time-series events with respect to alarms, incidents, failures, root causes, or other critical business events. This information constitutes a very long feature vector, changing continuously as events evolve in a sub-second to few seconds time scale, which needs to be mathematically processed in the form of mixed-nature in-memory and transient model banks [1]. Such a system is designed with several concepts in mind: online versus batch learning will mean that either complete models or transiently "relative-feature" mixed-nature model banks evolve continuously for up to several hours, while complete machine-learning-model distributions will be trained in prior, scheduled batch-mode runs and are, once available, completely frozen; new-service time may occur unboundedly close to older-service time; an operating-configuration complexity metric evaluated on the distributed model-storage system influences model-choice time materially; scale with feature count, alert volume, event volume, and other operational characteristics associated with contemporary architectures; drift affecting very rapid model-deployment times typically varies by the need for further operational choices; unwanted spurious detection of root causes of alerts can involve significant resources; and formal separation between monitoring for drift as opposed to service-evolution as input to model-selection processes is feasible [2].

III. LIMITATIONS AND CHALLENGES OF REAL TIME RCA

A. Data Ingestion and Processing Challenges

Real-time root cause analysis (RRCA) represents a distinct problem within the field of big data. More specifically, the distinction lies in the real-time requirement, since RRCA is feasible offline. Techniques developed for offline analysis may be applicable, but real-time solutions impose stricter constraints. RRCA typically requires the processing of time-series data generated by the target stream, along with side-channel metrics and additional contextual data. Architectural foundations for RRCA systems are reviewed in Section 3.

With the increased complexity of modern systems and their underlying components, the volume of operational data available has also increased. For example, Amazon Web Services (AWS) metrics serve as the input to over 700 operational dashboards. As system sophistication escalates, so does the complexity of their analyses. Consequently, organizations seek readily accessible and meaningful insights from their data. Operating with the already significant data volume of a target system may still be feasible. However, the inclusion of auxiliary sources, such as network telemetry, security events, product usage, and cloud-specific statistics, can inadvertently cause further complication.

Various other complications may be further noticeable. The delineation of primary from auxiliary sources can be difficult; is a dashboard modelling the time since the last database backup auxiliary or central? Many other components of the overall architecture should also fit this or a similar classification. Expanding auxiliary data architectures further complicates identification of the remaining rationales.

1) Data Latency and Throughput

With the wide availability of real-time streams such as monitoring logs, logs, auditing events, and quality incidents from telecom or cloud-based products, the demand for immediate diagnosis has become more pressing [1]. Static machine-learning models, adjusted to a snapshot of historical information, are ill-suited for the unpredictability of frequent pipeline or production changes. Furthermore, the frequency and volume of streaming data can reach levels that make batch computation infeasible; searching, reindexing, or retraining a model during ongoing diagnostic actions (potentially based on logs recorded just a second or two earlier) cannot maintain the necessary continuity [3]. At the other extreme, simply reengaging with a substantial training set still requires large amounts of time for extensive data preparation, and features obtained for offline or near-time systems in many cases do not apply [2]. Such needs present obvious general and technical implications regarding efficient feature engineering, model-design selection, diagnostic-action evaluation, analogy-based processes, and other arrangements on a data-record basis or in real time governed by sliding windows.

2) Data Quality, Provenance and Governance

Real-time root-cause analysis (RCA) relies on continuous, high-velocity streams of data from mission-critical applications and infrastructure. In such environments, data quality becomes even more critical than in traditional, batch-oriented big-data applications; if the system acts upon erroneous or erroneous data, the cost of analysis will increase, and the probability of addressing the actual root cause will decrease. In fact, inappropriate or irrelevant episodic big data due to erroneous data greatly hinders the system's ability to detect an event's true root cause, especially in real-time RCA [4]. Moreover, additional metadata regarding capture time, author, and location needs to be processed from the outset to avoid being lost during extraction from stream-based data sources.

Even a small number of missing or outlier values in a continuous data stream can misrepresent a significant portion of a facility's operating time, leading to misleading inferences about operating conditions [5]. Consequently, it is essential to maintain a consistent record of the data's provenance to track the physical source of any anomalies and understand their subsequent impact on the root-cause analysis. Attempting to recover the provenance of a data item without appropriate safeguards from the time of its acquisition would be extremely difficult.

In addition, RCA in real-time environments requires different safeguards from those deployed in non-real-time settings. Data arriving from different stream sources need to comply with pre-defined quality specifications to be ingested into the RCA pipeline. Continuously monitoring and managing the quality of the data flowing through different streams is essential to block the entry of tampered information. Enhancing the data-governance framework to address both of these issues remains a significant challenge.

3) Schema Evolution and Metadata Management

Almost all big systems need to cope with schema evolution because of source-system changes, internal-system redesigns, data re-processing needs or variations in the semantics of data flows, changes in the event-structure of data-sources, and other reasons [6]. Each schema evolution event typically mandates the adaptation of the downstream data-transformation logic. Even if all changes to data-sources are reflected in the associated metadata, such adaptation usually entails substantial effort and time. Data programs seldom capture data semantics and run queries against unnamed data and, regardless of the availability of enhanced metadata, data-program evolution support is thus scarce. Also, metadata capable of facilitating the understanding of potential impacts on downstream applications usually cannot reflect all dependency- and semantic-relation information in complex systems. Hence, existing approaches to schema-evolution adaptation do not provide sufficient assistance.

In a big-data system, metadata management assumes even greater importance because the high-volume variety of data requires careful elaboration of accurate metadata to help stakeholders concretely assess data-coverage availability or derive high-level information about the data. Metadata also proves essential for deciding the matching of data-flows, for documenting knowledge in the management of a data-science workflow, and for estimating costs associated with the running of operators and the storage of various materials. However, because of the highly costly and continuous nature of big-data operations and the typically limited human resources available, the processes of metadata collection, organization, and release often remain inefficient and unbalanced.

B. Feature Engineering for Real Time Diagnostics

Feature engineering plays a pivotal role in real-time root cause analysis. Since computations are typically performed on sliding time windows of the incoming data stream, the derived diagnostic features must be time-dependent and adjusted as new data arrives, ensuring smooth transitions when moving from one window to the next. These adjustments can often be implemented in a straightforward manner; for example, the rate of change of a signal can be computed by recomputing the maximum and minimum values within a time window and applying the global maximum and minimum observed thus far to avoid abrupt changes. Other, more complex features such as autocorrelation, multi-pulse detections, or wavelet coefficients tend to be less trivial to convert between fixed-window and sliding-window representations. Deliberately introducing periodicity into such features can provide clues about the expected time to failure, hence raising awareness about inspection scheduling; yet, such temporal information may sometimes indeed serve to distract decision-makers from the more fundamental issue.

When it comes to causality, there is no doubt that detecting causal relationships is of paramount importance; yet, in many applications causal relationship detection appears to be well approximated using measures addressing correlation only. The proverbial saying “correlation does not prove causation” nevertheless remains pertinent in many fields of expertise, one of them being the monitoring of large-scale software systems. Concluding that an observed correlation truly indicates causation, especially on the basis of a mere correlation coefficient can prove to be quite dangerous. Training decision-making models on a feature extracted from the past system state to predict the future attention one deserves invariably introduces the risk that medicative actions either preclude or trigger the anticipated attention. Developing a complex network of structural equation systems could help shedding more light on the relationships among the observations, yet sufficient expertise in the domain, not to mention considerable effort, remains necessary for its successful establishment. Therefore, striving to extract features from observations that carry both

direct and indirect indications of the future emphasis either without resorting to sophisticated causal models—or, at least, without constraining them to disregard direct consequences—constitutes a worthwhile objective.

1) Temporal Feature and Sliding window:

Temporal information about datasets, such as timestamps or elapsed time differences, is critical in real-time data analysis [1]. Time-based features can be defined using sliding windows or time bins. In the real-time root cause analysis context, time-based features must be determined from business knowledge and pre-computed together with the timestamp of the first log message from the session to which the log message belongs.

2) Causality Inference vs Correlation

Most machine learning algorithms excel at recognizing correlations among data but cannot discern causal relationships. On the contrary, determination of causality is essential to understanding the underlying mechanisms of the system. Various causal inference theories and techniques have been proposed, yet they typically focus on static datasets. Causal inference methods have been explored for several decades and are indispensable in genetics and biology. These methods infer a directed acyclic graph from observational data and can be categorized into constrained-, score-, function-, and gradient-based approaches. Established methods such as PC, GES, and LiNGAM face challenges including ambiguity in causal relations, extended computational durations, and stringent assumptions. Gradient-based methods introduced in the deep-learning era ameliorate some of these drawbacks, accommodating both linear and non-linear data, producing weighted directed acyclic graphs automatically, and finding application in medicine and biology. With directed acyclic graphs in hand, researchers can deploy graph algorithms such as breadth-first search, random walk, and PageRank to infer the root cause. Moreover, metric-based techniques enable automated, real-time root-cause localization that surpasses the performance of log- and trace-based approaches.
The models generate an abundance of candidate features, inviting spurious correlations that could mislead diagnosis if ill-judged features are included. Remedial actions apply causal inference to reduce false alarms. The very presence of a correlation does not imply causation; lunar halos appear before rainfall, creating a spurious correlation [7]. While causality closely resembles correlation, causal relations furnish deeper insights into the governing mechanisms of the system and therefore constitute clear objectives for feature selection.

3) Explainability and Model transparency

Real-time RCA generally involves machine-learning models to automate the analysis of diverse telemetry data and to identify the root cause of an incident [8]. Such models may be required to process continuous time-series data from heterogeneous sources [9]. When monitoring complex systems, the volume of incoming streaming data can exceed the system's data-processing capacity. Later sections highlight the need for carefully engineered features, illustrate different data-processing and streaming models, and discuss infrastructure-supporting choices, such as ensuring data-quality measures as part of the deployment phase. These design decisions also constrain selection of analytics platforms and are complementary to supporting the actionability of RCA.

C. Model Selection and Deployment in Streaming Environment

Typically, in static data settings, models trained off-line on historical data are later deployed to serve incoming data streams. The realization of such a pipeline in streaming environments gives rise to three challenges: (i) how to detect and eventually accommodate model drift over time, (ii) how to optimally exploit limited and transient computational resources, and (iii) how to control the emergence of system and model instances that replicate already deployed ones.

To address the first challenge, an agent detects the absence or presence of drifting phenomena in various aspects of the task undertaken by the model in production. The agent relies on different event detectors, including mechanisms not set in motion until a specified time has elapsed since the model deployment. Continuous derivatives of key statistics help characterize concept drift in unbalanced data, counteracting the need to adjust sampling or weighting strategies. Converging evaluations of the model's sampling quality complete the characterization of its working conditions (preamble stage). Periodic assessments secure that the agent does not overlook possible changes in the task without demanding a constant flow of feedback (surveillance stage). The collected observations enable the detection of functional drift; a set of meta-features describing the content of the incoming samples allows to reveal structural drift; the estimated model quality signals the potential irrelevance of the current model [1]. When the functionality of the system (i.e., the task it accomplishes) changes, the former model is deactivated and a new one is simultaneously sought to approach the "new" problem. When passage between independent tasks takes place, it is possible to keep the old model alive and pursue simultaneously the acquisition of a substitute for the "new" situation.

To meet the second requirement, the systems strive to constrain processes that early on generate new instances of a system that already exists in phase 2 of its life cycle or an alternative running instances characterized by identical preconditions and objectives. Upon satisfying the initial set of constraints imposed on the systems under consideration, they qualify their resource demands as low.

1) Online versus Batch Learning

In a streaming context, two main paradigms govern the training of predictive models: online and batch learning. Batch learning involves processing a series of homogeneous data chunks arriving at irregular intervals, while a more stringent interpretation restricts model updates to the arrival of entire data batches [10]. Despite being potentially appropriate for certain use cases, standard batch approaches break the obligatory real-time response loop required for root-cause diagnostics.

True online learning generation permits analysis of single observations via one-shot learning [11]. The suitability of various classifiers for each of these schemes varies considerably. Exploratory instance classifiers with low incremental cost, such as nearest-neighbor strategies and Local Outlier Factors (LOFs), select and label one instance at a time before issuing a verification request to a human19. Subsequently, the system tracks concept drift via distributional distance measures, ensuring relevance and preventing obsolescence.

2) Model Drift Detection and Adaptation

Application of machine learning (ML) is rapidly increasing, but, at the same time, challenges related to ML adoption are growing as well. Drift of model quality is one of the critical factors that affect the quality of ML applications over time [12]. Various scenarios can trigger drift, such as systematic changes in the environment, changes in processes or applications, and introduction of new classes. Advanced monitoring tools are crucial to tracking the model during operation, enabling effective detection of quality degradation and a smooth transition toward a new model when needed.

Assuming data and model distribution remains unchanged for the entire lifecycle of the ML application is a common misconception both in academia and industry. Monitoring of model performance on a new dataset is relevant after deployment. Moreover, monitoring of the input data itself is essential to spotting data drift, which occurs when the distribution of incoming data changes over time. A clear statistical difference between the training data and the serving data is directly related to the drop in model implementation ability [13].

3) Resource Management and Scalability

Data analytics systems operated within a cloud environment require constant monitoring of their resource utilization to govern how resources should be allocated. This is often a tedious task, however, as tenants

may need only simple forecasting methods to determine resource needs. Integration of the resource management techniques needs to be done for the automatically operated resource scaling to ensure an SLA. Request rates and several CPU-related metrics can be collected to estimate the time remaining until upscaling of the cloud service will be needed.
The cloud providers can avoid SLA breach by monitoring the central processing unit (CPU) utilization before an SLA breach occurs, taking advantage of the low correlation between this metric and other metrics. A multi-tier, repeating cycle model is presented. Multivariate time-series analysis of the historical observations detected significant cycles in request rates, which in turn were applied to the model.
Prediction methods such as artificially added multi-tier hierarchy, long short-term memory networks, and averages of several simple predictors can enhance accuracy in forecasting resource usage. Machine learning-based time-series forecasting is applied on cloud resources to determine maximum resource allocation before SLA breach for batch jobs [14].

D. Data Streaming and Infrastructure Challenges

The main challenge in building a real-time root-cause platform is not data velocity, but the need to satisfy tight error budgets while addressing a highly diverse set of use cases involving nearly 1000 features. Two additional challenges further complicate the problem: the need for fault tolerance and the requirement for observability to facilitate both proactive incident prevention and reactive incident response.
Fault tolerance is a fundamental property for streaming systems, ensuring that they function correctly despite failures in nodes, clusters, or external systems [1]. For root-cause platforms, exactly-once processing guarantees and the ability to recover from system-wide failures are essential [15]. Exactly-once guarantees prevent features from being computed more than once (e.g., in cases of replays due to downstream failures) or not at all (e.g., if the system fails between two successive operations). Such guarantees are particularly important for high-dimensional time-series data, as delays in the availability of one or more time-series features inhibit timely detection of upcoming anomalies.
Observability capabilities are critical. Without observability, developers lack visibility into the state of components and processes in the system and therefore cannot quickly resolve incidents such as prolonged outages or inaccurate alerts originating from production environments. Monitoring and observability enhancements for root-cause-analysis systems are designed to identify and locate minor error thresholds, proactively prevent further degradation, and determine root causes within the system.

1) Choice of Streaming Platforms

A real-time root cause analysis platform inevitably deals with different types of streaming data about system components, users, network, and applications. The choice of data streaming platforms has a great impact on the architecture and implementation design of a real-time root cause-analysis platform. Depending on the experience, knowledge, and requirement specifications from the industry partners, different data streaming platforms will influence the selection of other associated tools in either the ingestion, processing, or storage tier.
To illustrate the potential choices regarding the selection of streaming platforms, as part of the design phase of the real-time root cause. It is a consideration among the considerations in choosing root cause analysis solutions. Real-time root-cause analysis operates in a data collection and knowledge-building phase before extracting meaningful features, patterns, and persistent knowledge. To generate alerts for system diagnostics, the focus is set on high-dimensional online data sources. Depending on the service period of different applications, their events follow specific temporal patterns to determine the envelope of future occurrence. Capturing the shift of temporal patterns in service duration to trigger preventive and corrective alerts can play a crucial role in maintaining system quality. A pattern-based and time-aware root-cause-analysis solution can support the preliminary phase of a root-cause-analysis engine.

2) Fault tolerance and Exactly-Once Processing

Many critical infrastructures and processes cannot tolerate data loss during ingestion, processing, and storage phases. Consequently, systems deployed to monitor these critical systems should enable a high availability mechanism to allow continuous analysis of incoming data streams and provide alerts when necessary [16]. Business processes in industries such as data centers, electricity grids, utilities, and airports are disrupted regularly, resulting in significant revenue loss. Failures such as a power surge that halted Delta Airlines military operations at Hartsfield Jackson Airport demonstrated the importance of predicting, diagnosing, and providing root cause analysis of such equipment malfunctions. Data from sensors, equipment health reports, and historical files can be effectively analyzed using advanced machine learning techniques and predictive analytics technologies. A predictive analytics platform can therefore assist in improving system resilience, reducing downtime, and enabling root cause analysis.

3) Monitoring, Observability and Incident Response

Monitoring and incident response aim to identify and address the underlying causes of abnormal behavior in software applications [17]. Anomalies arise from the joint operation of a multitude of services, making such analysis inherently complex and intricate. Automated systems can help alleviate some of this cognitive burden. However, the interdependencies between services create challenges in determining which data to record, which surfaced symptoms to analyze, and which countermeasures to apply [18].

E. Real-Time Alerting, Visualization and User Interaction

Detecting anomalies and subsequently identifying their probable root causes is a complex endeavor that becomes even more difficult in a streaming environment. This section discusses mechanisms for improved (1) alerting and (2) user interaction to increase signal-to-noise ratios and enhance the overall user experience in an environment with potentially thousands of alerts or other generated items. Prioritizing alerts is essential [5]. A schema for potential alert types allows for streamlined monitoring of multiple information streams, and it clarifies the focus of individual alerts. Using alert shaping maximizes the utility of individual alerts by assigning alert categories, severity tiers, and priorities to facilitate further processing intelligently and inform additional visualization and narrative options in the user interface.

Narratives explaining the internal reasoning behind alerts, predictions, and the driving factors in underlying features can supplement the raw alerts provided [19]. Selecting appropriate graphical visualizations aids communication and attempts to convey insights in ways that resonate with the specific audience, avoiding overwhelming amounts of input. Collaboration capabilities facilitate human input to the analysis process, while monitoring logs of all agent decisions creates a semi-transparent audit trail for compliance verification and assurance of appropriate data and information handling.

1) Alert Prioritization and Noise Reduction

Given the increasingly complex and distributed nature of IT ecosystems, it is commonplace for IT organizations to be bombarded with multiple operational alerts, notices of application errors or exceptions, and failures across infrastructure components. While the number of issues may be considerably higher, very few investigations are typically launched [20]. Therefore, it is crucial to prioritize the incoming alerts and determine which incidents require immediate investigation.

An alert prioritization policy allows the real-time root cause analysis platform to assess and assign a priority score to every alert, taking its context—including alert type, affected entities, and accompanying metrics—into account along with a comprehensive transactional analysis of the dependencies and relevance of other ongoing alerts. When a new alert arrives, the score of its root cause is revised according to the most current knowledge. In addition, various noise-reduction techniques can be applied to filter out alerts that are part of an already addressed issue or identify the alerts driving the largest degree of user impact.

2) Root Cause Narratives and Visualization Design

The goal of root cause analysis is to identify the inputs, states, and actions that led to an undesired change in a system [21]. Analyzing such root causes pushes systems towards improved stability and performance. Causal diagram and narrative creation approaches plotted these causative relations in a broad array of settings. Such diagrams and statements describe happenings in a step-wise manner, usually tracing an eventual transition over various intermediate events. The graphical complexity at times overwhelms users with information. Less is usually more when crafting engaging visuals for important graphic design disciplines, and the same can be said for real-time root cause analysis.

Each event in a trace situated between a descriptor and the output often proves less relevant than the actual inputs of concern. A focus on inputs prioritizes access to the most suitable remedy instead of lingering on the undesired output. Narrative extraction technologies describe traces similarly, extracting infrequent yet high-demand corpus messages and writing concise English language recaps of large log archive verbs. Such devices guide users in grasping the core of their concern without a lengthy dig into the data. By transposing the original input-descriptor-output pathway into such widely adopted causal diagram and narration forms yet omitting the intermediary steps, the maintainer's earlier-seeded work [22] illustrates how balancing architecture outline expressiveness holds the potential to ease effective root cause analysis across considerable swaths of data.

3) Collaboration, Audit Trails and Compliance

At the end of the Root Cause Analysis (RCA) process, compositions summarize the causes that triggered the failure. When collaborating on problem determination, keeping a trail of audit logs to revisit the RCA and following incident response protocols helps remain compliant with regulations [5]. For instance, the Dodd-Frank Wall Street Reform and Consumer Protection Act require financial institutions to document changes in their systems thoroughly and maintain oversight of changes to their Code Base.

Enhancing the overall RCA experience is hence important, as a good experience boosts retention and interest in using the service [23]. To aid engineers in documenting the RCA narratives effectively, a narration composition technique draws inspiration from large language models [24]. User interface methodologies provide multiple alternatives to aide filtering, sorting, and viewing the RCA.

F. Data Privacy, Security and Privacy Considerations

Data privacy, security, and regulatory compliance are paramount when implementing root cause analysis systems using big data and machine learning. Sensitive data risks exposure during collection, transmission, storage, and processing, and such exposure can violate privacy regulations and create legal liabilities. Root cause analysis platforms often analyze time-series data from various systems, generating event data that tracks service performance and user behavior. Such event data can contain user identifiers, location information, or other confidential data, necessitating stringent policies [5].

Security risks arise from deploying machine learning models that rely on private data. Such data can be leaked through inadvertent exposure of the model itself or through query-based attacks, where an adversary learns about user data by observing the model's predictions given particular sets of input data. In addition, root-cause-analysis machine learning techniques, including survival analysis and causal inference, often leverage sensitive training data. A model that benefits from this private data—whether an intermediate, auxiliary, or post-analysis model—would be particularly susceptible to information leakage.

G. Evaluation, Validation and Benchmarking

Evaluation, validation, and benchmarking face significant challenges. With real-time root-cause analysis, hosting environments can be dynamic; thus, many ML models undergo retraining or redeployment. Rigorous evaluation and validation play a crucial role in guaranteeing acceptable model quality. Data can change significantly, resulting in two forms of model drift: input concept drift, where the data distribution has evolved, and label concept drift, where the target distribution has shifted [25]. Continuous automated

evaluation tools that process production data streams and calculate real-time performance metrics help with drift detection, but they cannot quantify changes in quality relative to historical standards. Thus, a second challenge is to compare the current model's performance against its historical quality and determine whether model drift warrants a retraining action. When Ockam's data-collection service was actively configured, jitter in the system caused random failures, temporarily creating redundant outputs. The system is now treated as research software, with maintenance effort shifted to a competing framework, Kafka [5]. A large off-site audit recommended terminating it on the grounds of missed service-level objectives, obsolete technologies, and limited capacity. Consequently, retraining is carried out on both batch and streaming data, but malfunctions can go undetected. Slowly drifting production topics may still reach a significant drift level. The approach in such cases is to continue using the model in production and establish a secondary framework that captures and analyzes the batch history of production data on a separate infrastructure.

Real-time evaluation metrics. Deploying models in production often requires evaluating their performance on incoming data streams. Nevertheless, the straightforward adoption of a single-stream data pipeline for real-time metrics may not suffice in evaluating models undergoing periodic retraining. Models are routinely retrained and may even operate on different data streams, while the production system does not directly expose available historical batches. Several other strategies exist to evaluate data-driven systems after deployment. When batch data about the inputs, signals, and predictions is available, it can be used to analyze the performance of deployed models. Certain approaches characterize the quality of completion flows, paying attention to monitoring percentage on the input series and controlling for seasonality and holidays.

1) Real-Time Evaluation Metrics

A real-time root cause analysis (RCA) platform infers and explains the reasons behind a system's failure or abnormal behavior using big-data analytics and machine learning. Evaluating the platform's performance presents numerous challenges, arising from the nature of streaming data and the required real-time operation of the system under evaluation. Video stream processing, ensuring both data latency and model performance with time-critical data, introduces multifaceted difficulties [5].

Real-time evaluation metrics must accommodate diverse data formats, such as variables, timestamps, and path information; extend through weeks or months, depending on urgency; and provide knowledge even when the root cause is already known to allow for continual validation and refinement of the analysis approach. Additional variables comprise noise filtering and various other properties of the time window not applicable in a static context. Infrequently occurring events or those with prolonged time separation pose substantial risks. Recognizing these needs and utilizing adequate evaluation techniques is essential for the effective assessment of a real-time RCA platform

2) Benchmarking Against Static Baselines

Root cause analysis (RCA) is the systematic identification of events, from complex and interrelated systems, driving the occurrence of a negative effect. The classical remedy for each occurrence of a 'failure' is a thorough investigation applying RCA, once a damaging incident has happened and the operational condition is still active. In contrast, proactive and online monitoring platforms aim to analyze current situations that may potentially lead to unpredicted breakdowns and perturbations. Therefore, RCA may occur in parallel with monitoring across the systems.

The transitional process from static to streaming data remains an open research problem for big data and machine learning approaches. Existing real-time monitoring services such as Azure Monitor, Google Monitoring, Grafana, and Prometheus primarily meet needs on a static basis. A clear distinction is that monitoring signals are regularly and uniformly sampled as a timeline, while alarms or alerts, generated from dynamic monitoring upon drifting detections and generated respectively either by users or automatic systems, is non-regular, sparse and unsynchronized across multi-class targets. Beside traditional and

conventional time-series attention-based or RNN models, additional exploration and research for asynchronous and irregular signaling modelling is valuable and interesting.

3) Deploying controlled experiments in Production

Designing interventions and determining their success constitute a fundamental goal in various applications, enabling online experimentation and gradual improvement of systems. Because many real-time systems seek to augment or automate such process, deploy the corresponding components as the deployment strategy for root cause analysis is often similar to that for interventions; these components are barely operational, however, and thus do not train a fully functional candidate. Even once functional, components still require interventions through careful analysis of multiple features; such analysis evades the scope of the approach. Nevertheless, revisiting root cause analysis in the context of layered systems allows deploying most of the original work without further development.

The problem of root cause analysis can thus still address a broader audience without counting on complete operability and can assist systems already equipped to deploy interventions. Strategies are available to mitigate complexities arising from a separated analysis. Retraining models according to widely varying characteristics of actively deployed components is likewise avoidable, focusing attention on other options. Organizing analysis requests differently substantially eases issues related to state analyses. Addressing an analysis request through either a set of available components or a deployment of the intervention under consideration narrows the search dramatically.

Maintaining broader relevance when continuous intervention systems remain unfeasible and limiting the additional development required ensure root cause analysis fits within broader experimentation and exploration plans. Hence can deploy a preliminary version in practice and address both previously surveyed and broader audience needs, whereas the earlier configuration broadens the analysis toolset without demanding substantial additional resources [13] [26].

IV. CONCLUSION

Real-time root cause analysis for multi-stream, high-velocity data remains challenging, with significant barriers still to be overcome despite advancements in machine learning and associated technologies. The architecture must address low-latency and high-throughput data ingestion and processing, including the need for on-the-fly metadata management, provenance tracking, and data governance. Feature engineering should permit the extraction and monitoring of temporal features from widely used communication protocols, the construction of causation-oriented features from event data, and the integration of explainability techniques to improve transparency without sacrificing performance. Various model training and selection strategies need consideration, including correlation-based model selection, approaches to drift monitoring and adaptation, and support for ensemble methods and carbon-aware optimization. The infrastructure design must support adequate observability, automated fault-recovery mechanisms, capturing of low-level yet semantically meaningful telemetry, and full integration with existing secured access facilities. Real-time user interaction capabilities remain critical, along with appropriate systems to prioritize alerts, generate rich and interactive causal narratives, support collaborative analysis, and provide detailed audit trails for compliance purposes. Data compliance processes require embedding into the deployment pipeline functionalities to monitor and classify real-time data records against regulatory frameworks, establishing a clear handover from model performance to privacy assessment, and monitoring and classifying documents after the extracted information is automatically processed. Evaluation systems still need to make progress by extending existing frameworks to accommodate streaming settings, isolating performance degradation on static benchmarks allowed in real-time scenarios, and conducting controlled experiments on production systems without disrupting users. Organizational and process-related hurdles complicate the development process, including the definition of root-cause topics, the extent of available telemetry, precise procedure documentation, and the specification of operational boundaries, with a clear understanding required of the interaction between

root-cause analysis and causality identification for cross-technology alerts. The level of applicable telemetry varies across systems, making collaboration a vital condition for providing alerts across multiple technologies on a single platform and the definition of succinct, expressive, and unambiguous deployment targets.

REFERENCES:

[1] S. Ge, H. Isah, F. Zulkernine, and S. Khan, "A Scalable Framework for Multilevel Streaming Data Analytics using Deep Learning," 2019.
[2] P. Basanta-Val, N. Audsley, A. Wellings, I. Gray et al., "Architecting Time-Critical Big-Data Systems," 2016.
[3] D. Bickson, G. Gershinsky, E. N. Hoch, and K. Shagin, "A Statistical Approach to Performance Monitoring in Soft Real-Time Distributed Systems," 2009.
[4] C. Chung and D. A. Jaffray, "Cancer Needs a Robust "Metadata Supply Chain" to Realize the Promise of Artificial Intelligence," 2021.
[5] E. Steven Kirkendall, Y. Ni, T. Lingren, M. Leonard et al., "Data Challenges With Real-Time Safety Event Detection And Clinical Decision Support," 2019.
[6] Z. Wang, L. Zhou, A. Das, V. Dave et al., "Survive the Schema Changes: Integration of Unmanaged Data Using Deep Learning," 2020.
[7] C. Zhihao and Q. Hongchun, "Review on Causality Detection Based on Empirical Dynamic Modeling," 2023.
[8] R. Vishnampet, R. Shenoy, J. Chen, and A. Gupta, "Root Causing Prediction Anomalies Using Explainable AI," 2024.
[9] A. Maged, S. Haridy, and H. Shen, "Explainable Artificial Intelligence Techniques for Accurate Fault Detection and Diagnosis: A Review," 2024.
[10] R. Ramanath, K. Salomatin, J. D. Gee, K. Talanine et al., "Lambda Learner: Fast Incremental Learning on Data Streams," 2020.
[11] K. Malialis, C. G. Panayiotou, and M. M. Polycarpou, "Data-efficient Online Classification with Siamese Networks and Active Learning," 2020.
[12] M. Banf and G. Steinhagen, "Who supervises the supervisor? Model monitoring in production using deep feature embeddings with applications to workpiece inspection," 2022.
[13] B. Eck, D. Kabakci-Zorlu, Y. Chen, F. Savard et al., "A monitoring framework for deployed machine learning models with supply chain examples," 2022.
[14] S. Kardani-Moghaddam, R. Buyya, and K. Ramamohanarao, "Performance-Aware Management of Cloud Resources: A Taxonomy and Future Directions," 2018.
[15] M. Pal Singh, M. A. Hoque, and S. Tarkoma, "A survey of systems for massive stream analytics," 2016.
[16] B. Prasad Majumder, A. Sengupta, S. jain, and P. Bhaduri, "Fault Detection Engine in Intelligent Predictive Analytics Platform for DCIM," 2016.
[17] J. Soldani and A. Brogi, "Anomaly Detection and Failure Root Cause Analysis in (Micro)Service-Based Cloud Applications: A Survey," 2021.
[18] D. Roy, X. Zhang, R. Bhave, C. Bansal et al., "Exploring LLM-based Agents for Root Cause Analysis," 2024.