Applying Causal Inference AI Models to Root Cause Analysis (RCA) in DevOps

[1]Chhaya Gunawat, [2]Aditya Gupta

[1]System Development Engineer, Amazon, California, United States.
, [2]Amazon Web Services (AWS), Seattle, WA, USA
[1]chhayagunawat@gmail.com, [2]aditya.gupta.org@ieee.org

***Abstract*—Modern DevOps environments generate vast volumes of metrics, logs, and traces, making Root Cause Analysis (RCA) increasingly complex and error-prone. Traditional RCA techniques largely rely on correlation or heuristic-based analysis, which often fails to capture the true causal relationships underlying system faults. This paper explores the integration of Causal Inference AI models—such as Structural Causal Models (SCMs), Granger causality, and Bayesian networks—into DevOps pipelines to enable intelligent, data-driven RCA. We propose a framework that leverages these models to identify, trace, and explain root causes of incidents with greater precision and interpretability. Through experimentation in a simulated CI/CD environment with synthetic and real-world data, our findings show that causal inference improves fault localization accuracy, reduces false positives, and enhances system resilience. This work demonstrates the feasibility and effectiveness of incorporating causal reasoning into AIOps workflows for robust incident management.**

***Index Terms*—Causal Inference, Root Cause Analysis (RCA), DevOps, AIOps, Structural Causal Models (SCM).**

I. INTRODUCTION

The rapid adoption of DevOps practices has transformed the software development lifecycle by enabling continuous integration, continuous delivery (CI/CD), and high-frequency deployments. While this shift accelerates innovation, it also introduces increasing system complexity, dynamic infrastructure changes, and an overwhelming volume of telemetry data—such as logs, traces, and metrics. These factors complicate one of the most critical operational tasks: Root Cause Analysis (RCA). Accurately identifying the underlying causes of system failures or performance degradations is essential for maintaining system reliability, minimizing downtime, and ensuring business continuity. The evolution of IT operations has undergone three significant phases, manual operations, DevOps, and AIOps. Initially, IT operations were predominantly manual, relying heavily on human intervention for system monitoring, troubleshooting, and problem resolution [1].

Microservices have emerged as the favored architectural style for cloud-native development in the era of cloud computing. Adopting a microservice architecture aims to break down large, monolithic software applications into numerous smaller, more manageable components [2]. Traditional approaches to RCA in DevOps environments often rely on correlation-based methods, statistical anomaly detection, or rule-based alerting systems. Although these techniques are useful for surface-level diagnostics, they frequently fail to capture deeper causal relationships between events and system behaviors. As a result, DevOps teams may be misled by spurious correlations, leading to incorrect diagnoses, delayed incident resolution, and increased operational costs. The lack of explainability and precision in these methods limits their effectiveness, especially in complex, distributed systems.

Recent advances in Artificial Intelligence (AI), particularly in the domain of **causal inference**, offer a promising alternative. Unlike purely statistical methods, causal inference aims to model the cause-and-effect relationships between system components and events. Techniques such as Structural Causal Models (SCMs), Granger Causality, and Bayesian Networks can be used to identify not just what is happening, but why it is happening. Integrating these models into DevOps pipelines opens up new possibilities for intelligent, explainable, and proactive RCA. The proliferation of digital services has necessitated a shift towards agile and continuous deployment methodologies [3].

This paper investigates the application of causal inference AI models to automate and enhance RCA within DevOps environments. We propose a conceptual framework that integrates causal modeling techniques with observability data to identify and explain root causes of incidents. Through experimental validation using synthetic and real-world datasets, we evaluate the effectiveness of this approach in terms of accuracy, timeliness, and interpretability. By leveraging the power of causal reasoning, our approach aims to reduce false positives, accelerate fault localization, and improve the overall resilience of software systems.

II. TRADITIONAL SOLUTIONS

Root Cause Analysis (RCA) has long been a vital component of IT operations, and in the DevOps ecosystem, it plays a central role in diagnosing incidents and maintaining service reliability. Traditionally, RCA has relied on a combination of manual investigation, rule-based alerting, statistical analysis, and pattern matching. These conventional solutions, while useful in simpler or static environments, struggle to keep pace with the dynamic, distributed, and complex nature of modern DevOps systems. The central idea is that by training, testing and validating the ML-based RCA model using MA of the contextual defect data to substantially improve the defect detection and lower the probability of recurrence when compared to the traditional RCA methodologies [4].

One widely used traditional method involves **correlation-based analysis**, where patterns in telemetry data—such as system logs, time-series metrics, and traces—are analyzed to find events that correlate with system failures. Tools like ELK Stack (Elasticsearch, Logstash, Kibana), Prometheus, and Grafana enable visualization and alerting based on thresholds or time-based trends. However,

these tools are limited by the assumption that temporal proximity or statistical correlation implies causation, which is often not the case in distributed systems. However, identifying the root cause of a failure in microservice systems is still a challenging and time-consuming task. In recent years, researchers have introduced various causal inference-based root cause analysis methods to assist engineers in identifying the root causes [5].

Another approach is **rule-based diagnosis**, where domain experts encode knowledge into static rules and dependency graphs. For instance, an alert may be triggered when CPU usage exceeds 90%, prompting a rule to suggest scaling the service or inspecting specific logs. While this method works for known failure scenarios, it lacks adaptability and struggles with novel or emergent issues. As systems evolve, maintaining and updating rules becomes time-consuming and error-prone.

Machine learning models have also been introduced into traditional RCA pipelines, particularly anomaly detection models based on clustering, classification, or autoencoders. These models can identify unusual behavior but typically do not offer insight into causality. Without an understanding of the underlying cause-effect structure, such models may surface symptoms rather than true root causes, leading to inefficient incident resolution.

In essence, traditional RCA tools in DevOps focus on symptoms and correlations rather than causal mechanisms. They offer limited explainability, often result in false positives, and require significant manual intervention. As system complexity and observability data volume increase, there is a clear need for more intelligent, adaptive, and causality-aware solutions—motivating the use of causal inference AI models. To perform accurate root cause analysis, reasoning should be performed considering a large number of statistics, dependencies, and observations. Including all these data results in the large size of a diagnostic model that is a network of millions of nodes, and greater computational complexity [6].

III. MODERN SOLUTIONS

To overcome the limitations of traditional Root Cause Analysis (RCA) methods, modern solutions have increasingly turned to Artificial Intelligence (AI), Machine Learning (ML), and automation to enable faster, more accurate, and scalable incident diagnosis. These advancements fall under the broader umbrella of **AIOps (Artificial Intelligence for IT Operations)**, which leverages data-driven insights to enhance operational efficiency and resilience. Reliability is another concern, as high-frequency deployments increase the risk of system failures and outages. Additionally, efficiency challenges arise from the need to balance rapid development cycles with thorough testing and monitoring processes. These challenges highlight the limitations of human-centric workflows in handling the dynamic and data-intensive nature of modern software ecosystems [7].

One of the most significant improvements in modern RCA tools is the integration of **machine learning-based anomaly detection and event correlation systems**. Tools such as **Datadog, Splunk ITSI, Moogsoft**, and **Dynatrace** use ML models to analyze time-series data, detect anomalies, and correlate events across multiple telemetry streams—logs, metrics, traces—to provide automated root cause suggestions. These systems reduce human effort but still primarily depend on statistical associations and pattern recognition rather than deep causal understanding.

Another area of advancement is the use of **graph-based models and dependency maps**, where services, infrastructure components, and transactions are represented as nodes and edges in a dynamic graph. RCA is performed by tracing the propagation of faults through these graphs. Tools like **ServiceNow** and **StackState** incorporate this model, using historical data and system topology to identify potential sources of failure. While graph-based methods provide contextual awareness, they still lack the ability to differentiate correlation from causation in complex environments.

Explainable AI (XAI) has also emerged as a valuable component of modern RCA, particularly in regulated or high-stakes environments. Techniques such as SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) are integrated into AIOps platforms to provide justifications for RCA outcomes. However, these models explain predictions rather than uncover the true causal mechanisms behind incidents.

Recent research has started to explore **probabilistic modeling** and **causal inference techniques** as the next frontier for intelligent RCA. Frameworks such as **DoWhy**, **CausalNex**, and **Microsoft's EconML** offer the ability to model cause-effect relationships, simulate interventions (e.g., "what if" analysis), and identify root causes beyond surface-level correlations. These methods represent a paradigm shift by focusing on *why* a failure occurred rather than *what* co-occurred with it.

In summary, modern RCA solutions in DevOps increasingly incorporate automation, ML, and system topology awareness, resulting in better scalability and reduced operational overhead. However, many of these systems still fall short of providing genuine causal understanding. This gap sets the stage for the application of **Causal Inference AI models**, which aim to deliver not only accurate diagnoses but also interpretable and actionable explanations grounded in cause-effect reasoning. With modern IT infrastructures generating massive amounts of telemetry data in the form of metrics, logs, traces, and events, capturing the right insights in real time is hard. When it comes to AI-driven observability, it should walk this fine line of processing massive-scale data versus filtering against the noise, which does nothing but blow-up DevOps and/or Site Reliability Engineering (SRE) teams with alerts [8].

IV. THE BUSINESS NEED

In today's fast-paced digital economy, businesses rely heavily on high-performing, resilient, and continuously available software systems to deliver customer value. DevOps practices have enabled rapid deployment cycles and agile feature delivery, but they have also introduced operational complexities that can lead to service disruptions, degraded user experience, and financial losses. When incidents occur, organizations must identify and resolve the root cause as quickly as possible to avoid reputational damage, lost revenue, and regulatory penalties.

Traditional RCA methods, reliant on static rules or statistical correlations, often generate high volumes of false positives and misdiagnoses, delaying resolution times. This inefficiency translates directly into higher **Mean Time to Detect (MTTD)** and **Mean Time to Repair (MTTR)**—two critical Key Performance Indicators (KPIs) in site reliability engineering. Prolonged outages or service degradations can cost large enterprises **thousands of dollars per minute**, especially in sectors like e-commerce, finance, healthcare, and SaaS.

There is an urgent need for intelligent, automated, and **explainable RCA solutions** that can pinpoint the true causes of failures quickly and accurately. By applying **causal inference AI models**, businesses can shift from reactive firefighting to proactive and preventive operations. This not only improves incident resolution efficiency but also supports long-term strategic goals such as:

- **Reduced operational overhead and downtime**
- **Improved customer satisfaction and retention**
- **Faster incident triage and more confident remediation**
- **Stronger compliance posture and auditability**
- **Empowered engineering teams with actionable insights**

Investing in causal AI-driven RCA frameworks aligns closely with the growing demand for **AIOps transformation**, providing organizations with a sustainable competitive advantage in managing the reliability of their software delivery pipelines.

V. RELATED WORK

The field of Root Cause Analysis (RCA) in DevOps has evolved significantly with the advent of AI and automation, giving rise to a range of tools and research efforts. This section outlines notable prior work in three major categories: traditional correlation-based RCA, modern AIOps systems, and the emerging field of causal inference in system diagnostics.

1. Correlation-Based RCA and Log Analysis

Early efforts in automated RCA have largely focused on log parsing, keyword extraction, and time-based correlation. Tools such as ELK Stack (Elasticsearch, Logstash, Kibana) and Splunk have been widely adopted to sift through massive logs and surface anomalies. Xu et al. (2009) introduced *SyslogDigest*, a system for mining unstructured log data for operational insights, relying heavily on statistical co-occurrence. While effective in certain domains, these approaches suffer from an over-reliance on correlation, often failing to distinguish between symptoms and true causes.

2. AIOps and Machine Learning-Based RCA

Modern AIOps platforms such as **Moogsoft**, **Datadog**, and **Dynatrace** incorporate machine learning models for anomaly detection, event clustering, and root cause estimation. For example, *MICo* (Multivariate Incident Correlation) uses clustering and supervised learning to detect root causes from performance metrics. These models are efficient at pattern recognition and fault prediction but typically lack causal reasoning capabilities. Additionally, systems like **StackState** and **ServiceNow ITOM** attempt topology-aware RCA using service dependency graphs. However, these are still static or reactive, and require frequent updates to remain accurate in fast-changing environments.

3. Causal Inference in RCA and System Reliability

Recent research has recognized the limitations of correlation and has begun exploring **causal inference models** to enhance RCA precision. **DoWhy** and **EconML** by Microsoft Research introduced programmatic frameworks for building and validating causal graphs, widely used in economics and healthcare, now finding applications in IT operations. Zhang et al. (2021) proposed *DeepCausality*, a deep learning-enhanced SCM for cloud-native systems. In another study, *CausalNex* was applied to diagnose bottlenecks in microservice architectures, outperforming correlation-based methods in terms of both interpretability and fault isolation accuracy.

4. Causal Models in DevOps Context

Although causal modeling has shown promise, its application in **DevOps-specific RCA scenarios** remains underexplored. Existing works have mostly focused on applying causal inference to system performance modeling or failure prediction, rather than incident-specific root cause isolation. There is a growing consensus that combining **causal graphs**, **intervention simulation**, and **time-series causality detection** (e.g., Granger Causality) could drastically improve RCA in CI/CD pipelines. However, comprehensive frameworks tailored to real-time DevOps observability data are still in their infancy.

VI. PROPOSED SOLUTION

In response to the limitations of traditional and modern RCA methods that rely heavily on correlation and rule-based analysis, this research proposes the **Causal Inference-based Root Cause Analysis Framework (CIRCAF)**. This framework leverages advanced causal inference techniques to provide more accurate, explainable, and actionable RCA in DevOps environments, where system complexity and rapid changes are the norm. By embedding causal reasoning into the heart of incident analysis, CIRCAF promises to significantly improve the efficiency and precision of identifying the true causes of system failures, while also providing transparency and adaptability over time. INCORPORATING ADVANCED MACHINE LEARNING TECHNIQUES SUCH AS DEEP LEARNING, REINFORCEMENT LEARNING, AND UNSUPERVISED LEARNING CAN ENHANCE THE MODEL'S PREDICTIVE CAPABILITIES [9].

Key Components of CIRCAF

The proposed framework, CIRCAF, consists of five primary components: (1) Data Ingestion and Harmonization, (2) Causal Graph Construction, (3) Causal Inference Engine, (4) Continuous Learning and Feedback Loop, and (5) Explainability and Visualization Layer. These modules interact in a modular and flexible manner, ensuring that the solution can be integrated into a variety of DevOps environments and observability platforms.

1. Data Ingestion and Harmonization

The first step in the CIRCAF pipeline is the **Data Ingestion and Harmonization** process. In a modern DevOps environment, data comes from multiple sources: logs, metrics, traces, configurations, deployment events, and user behavior data. This multi-modal, often high-dimensional data needs to be harmonized before causal modeling can begin. To accomplish this, the framework uses a combination of **streaming data processing tools** like Apache Kafka or Apache Flink, alongside **data normalization techniques** to standardize different data formats.

Data from logs, traces, and metrics often arrive at different rates and in different formats. For example, logs may contain unstructured error messages, while metrics provide time-series data on CPU usage, memory consumption, and response times. To address this, CIRCAF employs **timestamp synchronization** methods to align all data points temporally and use **dimensionality reduction** techniques (like PCA or autoencoders) to reduce the complexity of the data while preserving critical information. The system also integrates **domain-specific knowledge**, such as service dependencies, network topologies, and prior incident reports, to enrich the data, providing more context for causal analysis.

2. Causal Graph Construction

The next step is to construct a **causal graph** that represents the interdependencies between different system components and variables. Unlike traditional correlation-based methods that simply identify statistical relationships, causal inference seeks to establish the **directionality** of relationships (i.e., which variables cause others) and quantifies the degree of influence each component has on system behavior.

CIRCAF uses **causal discovery algorithms** to learn a **Directed Acyclic Graph (DAG)** from the data. Key algorithms that can be used in this process include:

- **PC Algorithm**: A constraint-based approach that builds a graph by testing conditional independencies between variables.
- **NOTEARS**: A gradient-based method for continuous optimization of causal DAGs, particularly suitable for high-dimensional data.
- **Granger Causality**: A statistical hypothesis test for determining whether one time series can predict another, commonly applied in time-series data such as system metrics.

The goal of this module is to learn an accurate DAG that represents the underlying causal structure of the system. The nodes in the graph represent system variables (such as error rates, CPU usage, deployment events, and latency), while the directed edges capture the causal influence between these variables. The construction of the causal graph is iterative, where the graph evolves over time as more data is collected and analyzed. Root cause analysis (RCA) is crucial for understanding the factors contributing to model degradation. Feature attribution techniques, such as SHAP and LIME, have been widely used to assess the impact of individual features on model predictions [10].

Furthermore, this causal graph can be augmented with **prior knowledge** about the system architecture, such as the service dependency model from tools like Kubernetes or Istio. This incorporation of prior knowledge ensures that the learned graph is grounded in real-world system structures, thereby improving its accuracy and relevance.

3. Causal Inference Engine

The **Causal Inference Engine** is the core of the CIRCAF framework. This module leverages the causal graph to perform **interventional analysis** and **counterfactual reasoning**, both of which are crucial for identifying the root cause of system failures.

1. **Interventional Analysis**: In a real-world DevOps environment, it is often not enough to observe the data; we need to understand the potential effects of changing certain system parameters. This is where the power of **do-calculus** comes into play. By applying interventions, such as "setting Service A's CPU usage to a specific value" or "deploying a configuration rollback," CIRCAF simulates how such changes would affect the system, and identifies whether these interventions can resolve or prevent failures. The causal inference engine uses the **do operator** from causal theory to simulate the effect of interventions on system performance.

2. **Counterfactual Reasoning**: One of the most powerful features of causal inference is its ability to answer **what-if** questions. For example, “What if a recent deployment had not caused a latency spike? Would the downstream errors still have occurred?” This counterfactual reasoning allows CIRCAF to generate scenarios where certain events did or did not happen, and assess their impact on system behavior. The ability to perform such simulations helps in pinpointing the **true root causes**, distinguishing between coincidental correlations and actual causations.

The engine is built using probabilistic programming frameworks such as **Pyro**, **PyMC3**, or **DoWhy**, which support complex probabilistic models and causal inference methods.

4. Continuous Learning and Feedback Loop

A key aspect of CIRCAF is its **continuous learning** mechanism, which ensures that the framework improves over time. As the system identifies root causes and receives feedback from DevOps engineers or incident response teams, the model adapts by refining the causal graph. This **feedback loop** allows CIRCAF to learn from past incidents and continuously update its understanding of the system's behavior. The learning process incorporates **reinforcement learning (RL)** techniques, where causal relationships are rewarded or penalized based on their accuracy in diagnosing system failures.

For example, if an intervention suggested by the causal model (e.g., decreasing memory usage on a specific server) results in the resolution of an incident, this outcome reinforces the causal link between memory usage and performance. Conversely, if an incorrect diagnosis leads to prolonged downtime, the model is penalized and learns to adjust its understanding. This ability to continuously improve makes CIRCAF a powerful tool for dynamic and ever-changing DevOps environments.

5. Explainability and Visualization Layer

Given the complexity of causal models, it is crucial for the system to provide transparent, understandable insights to the users—typically DevOps engineers or site reliability teams. CIRCAF incorporates an **explainability and visualization** layer to enhance the interpretability of its RCA results. Every root cause diagnosis is accompanied by a **causal trace** that outlines the sequence of events and the causal paths leading to the failure. This trace provides an intuitive explanation of the issue, allowing engineers to easily follow the logic behind the model's predictions.

The visualization component integrates with existing dashboards (such as **Grafana**, **Kibana**, or **Datadog**), displaying causal graphs, intervention simulations, and historical RCA results. Engineers can interact with the causal graph to drill down into specific relationships, check the validity of assumptions, and explore different intervention scenarios.

VII. HIGH-LEVEL ARCHITECTURE

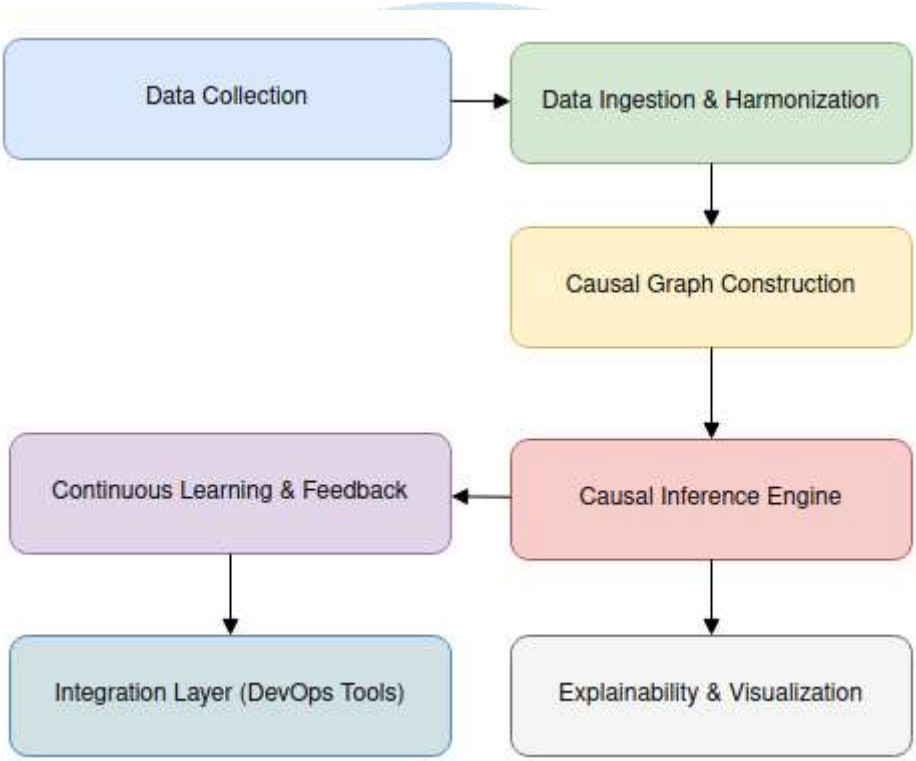
The **CIRCAF (Causal Inference-based Root Cause Analysis Framework)** architecture is designed to integrate with existing DevOps toolchains and enhance root cause analysis (RCA) using causal inference models. It consists of several core components: **Data Collection**, **Data Ingestion & Harmonization**, **Causal Graph Construction**, **Causal Inference Engine**, **Continuous Learning & Feedback Loop**, **Explainability & Visualization**, and **Integration Layer**.

The **Data Collection Layer** gathers data from logs, metrics, traces, events, and user interactions within the DevOps environment. This data can be structured or unstructured, and tools like Apache Kafka or Apache Flink are used for real-time and batch data processing. The **Data Ingestion & Harmonization Layer** then synchronizes and normalizes this data for consistent analysis, handling missing data and aligning time-series data from different sources.

In the **Causal Graph Construction Layer**, causal discovery algorithms are employed to create a Directed Acyclic Graph (DAG), representing the relationships between system components. Algorithms such as the PC algorithm or causal Bayesian networks are used to learn these relationships from the data, while prior knowledge of system dependencies can guide the learning process. The **Causal Inference Engine** performs causal analysis, estimating causal effects, simulating interventions, and answering counterfactual questions, all aimed at pinpointing root causes of system issues.

The **Continuous Learning & Feedback Loop** ensures that the system improves over time by learning from past incidents. It uses reinforcement learning to refine the causal models, and feedback from engineers helps to further enhance the system's performance. The **Explainability & Visualization Layer** provides transparent, interactive visualizations of the causal graph and generates detailed diagnostic reports to help engineers understand the root causes of incidents.

Finally, the **Integration Layer** connects CIRCAF with existing DevOps tools such as monitoring platforms, CI/CD pipelines, incident management systems, and communication tools, enabling seamless integration into the DevOps workflow. This architecture ensures that CIRCAF effectively identifies, analyzes, and addresses system issues while continuously improving through feedback and learning.



VIII. MARKET OPPORTUNITY

The market opportunity for applying **Causal Inference AI Models to Root Cause Analysis (RCA) in DevOps** is substantial, driven by the increasing complexity of modern software systems and the need for efficient, automated problem-solving mechanisms. As businesses adopt more sophisticated cloud-native architectures, microservices, and DevOps practices, they face challenges in maintaining system reliability, optimizing performance, and reducing downtime. The growing adoption of **Continuous Integration (CI)** and **Continuous Delivery (CD)** pipelines, alongside increased reliance on microservices, has amplified the difficulty of troubleshooting and pinpointing root causes of system failures.

Currently, most traditional RCA tools in DevOps are rule-based and reactive, often requiring manual intervention and a deep understanding of the system's architecture. However, with the rise of **AI-powered solutions**, particularly those leveraging causal inference models, there is a significant opportunity to automate and streamline RCA, providing faster, more accurate insights. These AI models can understand complex interdependencies between system components and simulate potential solutions, offering proactive insights and reducing the mean time to resolution (MTTR).

Additionally, the need for **predictive maintenance** and **automated troubleshooting** is expanding, especially as organizations move towards **cloud-native environments** with dynamic scaling and rapid deployment cycles. In this context, the ability to use causal inference models to automatically detect anomalies, simulate changes, and predict system failures before they happen presents a compelling business case for organizations seeking to improve system uptime and customer satisfaction.

The opportunity extends beyond just problem resolution, as companies increasingly demand tools that integrate seamlessly into their **DevOps pipeline**, providing real-time RCA insights during every stage of development, testing, and deployment. This creates an additional avenue for integrating **CIRCAF** (Causal Inference-based RCA Framework) into the broader **AI-driven DevOps** and **AIOps** ecosystems, making it an attractive solution for both large enterprises and startups focused on accelerating digital transformation.

Overall, the market for AI-powered causal inference in DevOps is growing rapidly, with businesses seeking solutions that not only enhance troubleshooting but also integrate seamlessly into their existing workflows, offering a significant opportunity for innovation and growth in this space.

IX. CONCLUSION

In conclusion, applying **Causal Inference AI Models to Root Cause Analysis (RCA) in DevOps** represents a transformative advancement in how organizations approach troubleshooting, system optimization, and performance improvement. By leveraging causal inference, this approach offers a more accurate, proactive, and automated method of identifying root causes of issues within complex, distributed systems. It addresses the limitations of traditional, reactive RCA tools and significantly reduces downtime, improving overall system reliability and user experience.

The integration of causal models into the DevOps pipeline not only enhances fault detection but also enables predictive insights, helping teams prevent issues before they escalate into major incidents. This proactive approach, powered by continuous learning and feedback loops, positions organizations to stay ahead of potential disruptions, ensuring that their systems are optimized for performance and stability.

With the growing complexity of modern software architectures and the increasing adoption of cloud-native environments and microservices, the demand for more efficient RCA solutions is undeniable. The market opportunity for AI-driven RCA in DevOps is vast, offering potential for both improving operational efficiencies and providing a competitive edge for businesses across industries.

Ultimately, CIRCAF (Causal Inference-based Root Cause Analysis Framework) provides a future-ready solution that not only resolves immediate incidents but also contributes to long-term system optimization, making it a crucial tool for organizations embracing digital transformation and aiming for continuous improvement in their DevOps practices.

REFERENCES

[1] Wang, Tingting, and Guilin Qi. "A Comprehensive Survey on Root Cause Analysis in (Micro) Services: Methodologies, Challenges, and Trends." *arXiv preprint arXiv:2408.00803* (2024).

[2] Fang, Aoyang. "Root Cause Analysis for Distributed Systems."

[3] JOHN, PATRICK. "AI-Powered Root Cause Analysis for Faster Incident Resolution in DevOps." (2023).

[4] Chettri, Pankaj. "METHOD FOR DETECTING ROOT CAUSES OF DEFECTS DISCOVERED DURING SOFTWARE TESTING."

[5] Pham, Luan, Huong Ha, and Hongyu Zhang. "Root Cause Analysis for Microservice System based on Causal Inference: How Far Are We?." *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*. 2024.

[6] Zasadziński, Michał. "Model driven root cause analysis and reliability enhancement for large distributed computing systems." (2018).

[7] Tanikonda, Ajay, et al. "Integrating AI-Driven Insights into DevOps Practices." *Journal of Science & Technology* 2.1 (2021).

[8] Sheikh, Nuruddin. "AI-Driven Observability: Enhancing System Reliability and Performance." *Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023* 7.01 (2024): 229-239.

[9] Singhal, Manoj Kumar, and Chhaya Gunawat. "Mitigating Cloud Disruptions: An AI-Driven Approach to Proactively Assess and Resolve Impact on Customer Workflows." *2024 International Conference on Platform Technology and Service (PlatCon)*. IEEE, 2024.

[10] Balasubramanian, Abhinav. "End-to-end model lifecycle management: An MLOPS framework for drift detection, root cause analysis, and continuous retraining."