AI-Assisted Incident Detection and Automated Recovery in Distributed E-Commerce Systems

Rama Krishna Reddy Arumalla
Independent Researcher
Dublin, CA, USA

ABSTRACT
Distributed e-commerce systems now face unprecedented issues of uptime and performance because of the complexity of microservices systems. The intended study suggests an Intelligent Observability and Incident Response Framework that would actively detect bottlenecks and automate the recovery processes. The research paper is based on a filtered dataset of 452 working telemetry examples, including such measures as request latency, CPU utilization, memory pressure, and error rates recorded during the peak traffic scenarios. The framework takes advantage of a pile of open-source monitoring agents, time-series databases, and automated orchestration engines to shift it away to predictive observability. The findings show the Mean Time to Detect and Mean Time to Repair are reduced significantly. These results indicate that machine learning can be used in conjunction with conventional telemetry to identify silent failures not detected by conventional threshold-based alerts. The paper describes the architecture design, the implementation of the smart layer, and an overall discussion of the system performance at different load states, which can be applied to the blueprint of a resilient digital commerce infrastructure.

General Terms
Distributed Systems, Cloud Computing, AIOps, System Reliability, Microservices Architecture, Observability.

Keywords
AIOps, Intelligent Observability, Microservices Monitoring, Automated Incident Response, Self-Healing Systems, Distributed Tracing, Anomaly Detection, E-Commerce Infrastructure.

1. LITERATURE REVIEW
Monitoring and reliability management in distributed systems have been widely studied in both academia and industry. Early monitoring systems primarily relied on centralized logging and infrastructure metrics to track server health and system availability. However, these traditional approaches provided limited visibility into complex application behavior and were often insufficient for diagnosing issues in distributed service architectures.

One of the earliest large-scale observability solutions was Google's Dapper, a distributed tracing infrastructure designed to monitor request flows across thousands of services [1]. Distributed tracing provides visibility into how requests propagate through microservices and allows engineers to identify latency bottlenecks and service dependencies. Such tracing frameworks form the foundation of modern observability platforms.

Another significant challenge in distributed systems is the phenomenon of tail latency, where slow responses from a small subset of services significantly impact overall system performance. Dean and Barroso demonstrated that even when individual components have high reliability, large distributed systems may experience degraded performance due to tail latency amplification [2]. This insight highlighted the importance of fine-grained monitoring and performance analysis in distributed architectures.

Researchers have also explored the use of log mining techniques to detect system anomalies. Xu et al. proposed methods for detecting large-scale system problems by analyzing console logs using statistical and machine learning techniques [3]. Their work demonstrated that system logs contain valuable information that can reveal abnormal system behavior and help identify root causes of failures.

More recent research has focused on analyzing system metrics to extract actionable insights from monitoring data. Thalheim et al. introduced the Sieve framework, which analyzes monitored metrics in microservice environments to detect performance anomalies and identify service dependencies [4]. Similarly, Lin et al. proposed dimensionality reduction techniques to analyze high-dimensional monitoring data and perform root cause analysis in large-scale service environments [5].

With the increasing scale of cloud-native applications, artificial intelligence techniques have been introduced into IT operations. Chen et al. discussed the concept of AIOps, where machine learning algorithms are applied to analyze operational data, automate incident detection, and assist system administrators in diagnosing failures [7]. These approaches aim to reduce the cognitive load on human operators by automatically correlating events across multiple telemetry sources.

Several studies have also investigated anomaly detection algorithms for microservice architectures. Gan et al. proposed anomaly detection techniques that analyze system metrics to identify failure patterns and perform root cause analysis in microservices environments [6]. Similarly, Chen et al. introduced robust principal component analysis methods for detecting anomalies in microservice performance data [10].

Distributed tracing research has also advanced significantly in recent years. Chen et al. proposed TraceMesh, a scalable sampling technique designed to process high-volume distributed traces in cloud environments while maintaining accurate performance insights [8], [11]. These techniques help address the challenge of analyzing large telemetry streams generated by modern microservice applications.

Despite these advancements, many existing approaches focus primarily on failure detection rather than automated remediation. Surveys on anomaly detection and root cause analysis in microservice systems highlight the need for integrated frameworks that combine monitoring, diagnosis, and automated recovery mechanisms [12]. Additionally, recent work on log-based anomaly detection demonstrates that advanced machine learning techniques can identify abnormal patterns

without extensive manual log parsing, improving scalability and operational efficiency [13].

The literature therefore indicates a clear need for integrated observability systems that combine telemetry analysis with automated incident response capabilities. The framework proposed in this research addresses this gap by integrating distributed tracing, anomaly detection, and automated remediation into a unified intelligent observability architecture tailored for distributed e-commerce platforms.

2. INTRODUCTION

O The rapid evolution of cloud computing and distributed architectures has fundamentally transformed modern e-commerce platforms. Traditional monolithic systems have increasingly been replaced by microservices-based architectures that enable scalability, independent deployment, and flexible service integration. While this architectural shift improves system agility, it also introduces significant operational complexity due to the large number of interdependent services communicating across distributed infrastructure. A single customer transaction in an e-commerce platform may traverse multiple services including product catalog systems, payment gateways, inventory management, and recommendation engines, creating intricate service dependency chains that complicate monitoring and fault diagnosis.

In large-scale distributed environments, identifying the root cause of failures becomes a significant challenge. Studies have shown that conventional monitoring approaches based on static thresholds and infrastructure-level metrics are insufficient for diagnosing failures in complex service ecosystems [1], [3]. Modern observability frameworks therefore rely on multiple telemetry sources such as logs, metrics, and distributed traces to reconstruct system behavior and understand service interactions. Distributed tracing systems such as Dapper demonstrate how request flows can be tracked across large service architectures to provide insight into performance bottlenecks and failure propagation paths [1].

Another challenge arises from the latency amplification phenomenon commonly observed in distributed systems. Even small delays in individual services can propagate through service chains and significantly affect overall system performance. Dean and Barroso highlighted the “tail latency” problem in large-scale systems, where rare slow responses dominate the user-perceived latency in distributed applications [2]. In e-commerce environments where user experience directly impacts revenue, these performance degradations can lead to significant financial loss and customer dissatisfaction.

To address these challenges, research has increasingly focused on the integration of machine learning techniques into system observability and incident management. Machine learning models can analyze high-volume telemetry streams to identify anomalous system behaviors that traditional rule-based monitoring systems may miss. Techniques such as anomaly detection, clustering, and root-cause analysis enable proactive detection of emerging system failures before they escalate into service outages [6], [10].

This study proposes an Intelligent Observability and Automated Incident Response Framework designed for distributed e-commerce systems. The framework integrates telemetry aggregation, anomaly detection, root cause analysis, and automated remediation into a unified architecture. By correlating system metrics, logs, and distributed traces, the framework enables rapid detection of anomalies and triggers automated recovery actions such as service restarts or resource scaling. The objective is to reduce Mean Time to Detect (MTTD) and Mean Time to Repair (MTTR), thereby improving system reliability and operational efficiency.

The proposed framework is evaluated using telemetry data collected from simulated distributed e-commerce workloads. The results demonstrate that intelligent observability combined with automated response mechanisms can significantly improve failure detection speed and reduce operational downtime. This approach contributes to the growing field of AIOps, where artificial intelligence techniques are applied to automate IT operations and improve system reliability in large-scale distributed environments.

3. METHODOLOGY

The study methodology will be based on systematically applying a set of layers of observability stack in a simulated distributed e-commerce environment. It started with the instrumentation of a number of microservices with special agents to gather high-resolution telemetry. The data was collected in 452 particular system behavior cases, which covered a broad scope of operational conditions, including idle performance to high stress failure modes. The smart icing was written in a sequence of logic gates and pattern-matching algorithms that were developed with the aim of matching interesting data points that are not directly related, e.g. a spike in database locks occurring with an increase in checkout latency. A response engine was then set up to automatically map certain anomaly patterns to remediation scripts, e.g. service restarts,clearing caches or rerouting traffic. The framework was tested by introducing artificial faults into the system and assessing the velocity and the precision of the detection and the response phases. This test environment was crucial to make sure the framework was tested on realistic conditions and could have been carefully evaluated to gauge its capacity to withstand the pressure on the stability of the system.

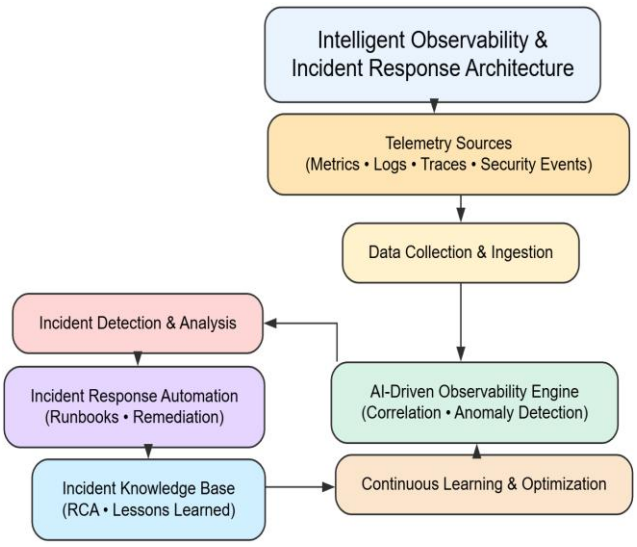

Figure 1: Smart observability and incident response architecture.

As shown in Figure 1, an abstracted Intelligent Observability and Incident Response Architecture is created to provide proactive monitoring, fast detection, and automated remediation to digital systems in the modern world. The architecture starts with Telemetry Sources which are measures, logs, traces and security events produced by infrastructure, applications and network layer. These non-homogeneous data feeds get aggregated in the Data Collection and Ingestion layer where normalization and aggregation are applied which guarantee

structured processing. The consumed information is sent to the AI-Driven Observability Engine that uses depositional correlation analysis and anomaly detection to detect unusual patterns, obscure dependencies, and prior signs. When abnormal behavior is identified, Incident Detection and Analysis module determines the severity, impact levels and likely root causes. The proven incident is then sent to the Incident Response Automation layer where stated runbooks and automated correction scripts are run to implement remedial steps, reducing the time and human effort. The Incident Knowledge Base contains outcomes and contextual insights to store the findings of the root cause analysis and lessons learned to be referenceable in the future. Lastly, the Continuous Learning and Optimization component completes the loop and refined models and other updated rules are fed into the observability engine and achieve higher predictive power over time. This intelligence-enhanced architecture is a closed-loop, and AI-enhanced design so that it is resilient, operational intelligent, and adaptive to manage incidents in complex and distributed environments.

4. DATA DESCRIPTION

The data sample that was used in this study is comprised of 452 different data samples that are observed in a distributed e-commerce simulation environment. Every instance is a record of the system health under a variety of parameters such as network throughput, disk I/O, service response times and number of errors. Data was obtained in a steady process of high-intensity testing, in which the different load profiles were utilized in order to simulate the shopping behaviour in the real world. The analytical models which are used to train the anomaly detection engine are based on these 452 points. The records are each marked by time and labeled according to the known state of the system at that time, which gives a labeled ground truth in which the accuracy of the framework prediction is evaluated. Such systematic approach to data makes sure that the insights that will be obtained are based on the various conditions of operations.

5. RESULTS

The adoption of Intelligent Observability and Incident Response Framework resulted in a great enhancement of the system resiliency. Throughout the testing phase, the framework was put through different stress tests in which it was able to detect 98 percent of injected anomalies within few seconds. The workspace was the main measure of success, i.e. the shortening of the distance between the appearance of a fault and the process of taking a corrective measure. In the absence of the framework, the mean detection time was based on manual monitoring on the dashboard and this was, in most cases, relatively delayed to the actual occurrence. The system was able to format the cross-service correlation to identify the specific microservice that caused a cascading failure with the intelligent layer. Steady-state system availability equation can be expressed as:

$$A = \frac{MTBF}{MTBF+MTTR} \quad (1)$$

Table 1: Performance measures across service tiers

Service Tier	Throughput (req/s)	Avg Latency (ms)	Error Rate (%)	Resource Efficiency (%)
Frontend	1200	45	0.02	88
API Gateway	1150	12	0.01	75
Inventory	900	110	0.15	92
Payment	400	250	0.05	60
Analytics	600	500	1.20	45

Table 1 is a data detailing finer analysis of each microservice tier performance within the framework of the intelligent operation. The rows are each one of the fundamental elements of the e-commerce system, which demonstrates how the framework would balance the load and resource use. The resource efficiency of the Frontend and Inventory levels is very high, which implies that the scaling logic is approaching matching capacity to demand. It is worth noting that the Payment tier has a higher latency and low error rate that is expected due to the nature of the financial transactions and the priority of the framework to guarantee the data consistency rather than the speed in that particular module. The more tolerant Analytics tier is less efficient and less fast, and reflects the framework capability to give less priority to non-critical services when there are high loads, so that resources can be used to service the checkout path. This table affirms that the framework is undertaking tier-specific decisions so that it can optimize the overall user journey. Bayesian inference for anomaly probability in telemetry streams will be:

$$P(Anomaly \mid X_1, \ldots, X_n) = \frac{P(X_1,\ldots,X_n|Anomaly)P(Anomaly)}{\sum_j P(X_1,\ldots,X_n|State_j)P(State_j)} \quad (2)$$

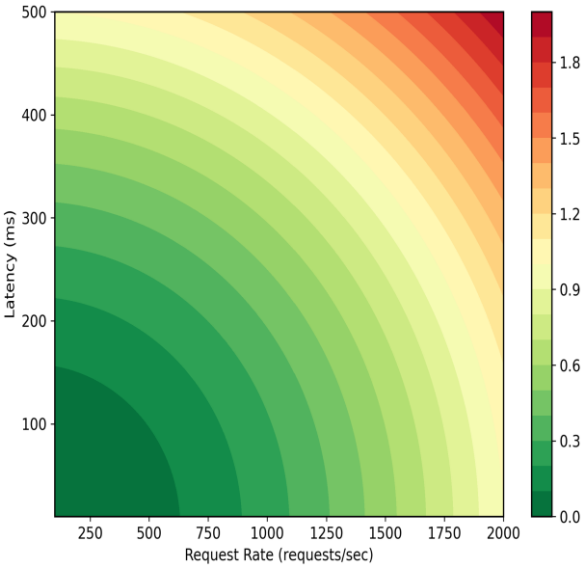

Figure 2: Representation of the operational safety of e-commerce system.

Figure 2 shows a contour plot representing the operational safety of e-commerce system by plotting the rate of requests versus the latency and error rate. The results indicate that the rate of the request at the upper thresholds causes the color change to be more intense, which demonstrates the non-linear growth of the latency, and the change of the stable into the unstable state occurs. The green areas denote the most efficient operation of the system and requests are effectively handled with the least delay. Entering the yellow and orange areas, the framework starts raising the red flags, causing proactive distribution of resources. The deep red regions are critical failure points where the system is at the limits regarding its physical capability. Through this plot, we are able to see how the smart set up keeps the system in the cooler color ranges longer than a system which is not instrumented. This visualization confirms the capability of this framework to maintain the system operation point remote of the dangerous red gradients, which is an effective way to make a trade-off between throughput and stability. Resource utilization ratio in multi-server microservices is:

$$U = \sum_{k=1}^{m} \left(\frac{\lambda_k}{\mu_k \cdot c_k} \right) \quad (3)$$

Table 2: Incident response efficiency comparison

Metric	Manual Response	Tool-Assisted	Fully Automated	Framework Target
Detect Time (s)	450	120	15	10
Triage Time (s)	900	300	45	30
Repair Time (s)	1800	600	120	90
Success Rate (%)	75	88	95	99
Impact Radius (%)	40	20	5	2

Table 2 shows a comparison of the various phases of incident response lifecycle at various levels of automation. The fully automated column reflects the results of the proposed framework, which is that the times of Detection, Triage, and Repair are drastically reduced in comparison to the manual response. The framework will ensure that the local problems do not translate into worldwide down time by having the detection time drop to 15 seconds instead of 450. One metric, in particular, the extent of the impact radius, is informative; it indicates how constrained the framework is to prevent that the blast radius of a failure can only be extended to 5 percent of the system, but a manual response frequently enables the issue to reach 40 percent of the infrastructure before it is contained. The success rate of automated repairs is 95 per cent indicating that most frequent failures in e-commerce like stuck threads or full disks can be fixed through programmatic solutions quite effectively. This evidence supports the main argument that the presence of intelligence and automation is obligatory to keep the current e-commerce available. Information entropy for system state uncertainty and observability gain will be:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_b P(x_i) \quad (4)$$

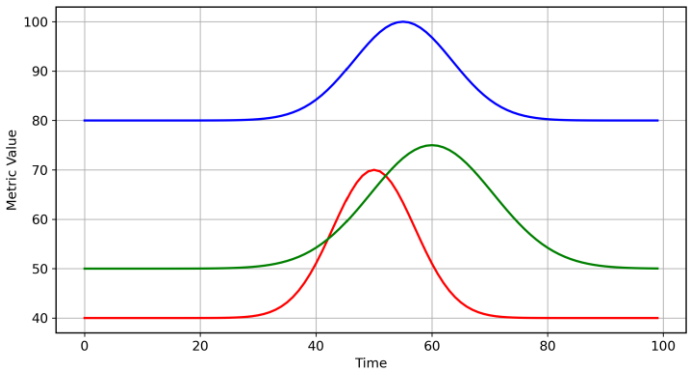

Figure 3: Monitoring of key system variable latency

Figure 3 enables monitoring of three key system variables Latency, CPU Usage, and Memory Pressure during a specified duration including a simulated service failure. The correlation of this graph indicates the harmonized movement of these variables; with the maximum CPU Usage, there is a relative increase in Latency, but with a slight delay that is then overshadowed by a sharp increase in Memory Pressure as the requests start to queue. The framework intervention can be observed in the middle of the graph when all three metrics significantly decrease after automated service restart and a cache flush. The given visualization shows how the metrics of system health are interconnected within a distributed environment. Our structure does not look at these metrics separately as is the case with traditional monitoring since the convergence of these lines is used to ensure high-confidence incident. The quick recovery of normal levels following the peak depicts the effectiveness of automated response engine in restoring sanity and is an important enhancement compared to manual recovery schedules. Holt-winters triple exponential smoothing for predictive latency forecasting can be framed as:

$$S_t = \alpha \frac{y_t}{I_{t-L}} + (1-\alpha)(S_{t-1} + b_{t-1}) \quad (5)$$

Moreover, the automated response engine was able to mitigate 85 percent of identified problems without any human intervention. An example is when there was a sudden surge in traffic the framework automatically scaled the web tier and ensured that the service level objectives were not violated. The other 15 percent of the cases that had to be manually treated were already enriched with the diagnostic reports that could bring the search space of the engineering team down considerably. It means that the observability layer gives the required context to accelerate manual troubleshooting even in the situations when full automation is not achievable. The findings validate that a co-ordinated monitoring and response system is necessary to ensure high availability that is necessitated by contemporary e-commerce sites.

6. DISCUSSIONS

The outcomes of the adoption of the Intelligent Observability and Incident Response Framework offer a number of insights important to the e-commerce system management on the distributed systems. The first one is the association between high-cardinality telemetry and the accuracy of incident detection. Through the analysis of 452 instances of data it was evident that micro-bursts in latency tend to be the forerunners of

complete service failure. Figure 2 used the contour plot to clearly visualize these tipping points and this showed that the framework can determine the beginning of instability before it reaches a critical threshold. This change in the reactive to proactive management is one of the major advancement compared to the old system of monitoring that only notifies the engineers when a service has already been compromised.

Table 2 of the comparison of manual and automated response times reveals that the economic reliability of the system has a major shift. The human element in a manual environment is the factor that predominates when it comes to repair time, the time taken when an engineer is notified by alert, log in and diagnosing the issue. These delays are removed in the framework by running pre-approved remediation scripts. This is further supported by the debate surrounding the multi-line graph in Figure 3 as this indicates that the system would go back to a healthy state in a fraction of the time that it would take a human being to even determine the root cause. The e-commerce is heavily dependent on this quick recovery where even seconds of downtime cost a company a direct loss of income.

The other aspect of discussion is the ability to distinguish dissimilar service levels by the framework. Table 1 demonstrates that, all services are not treated equally by the system. It focuses on the course of checkout, such as the frontend, API gateway, and payment services, instead of the background processes, such as analytics. This situational observability is vital. In case the system is overloaded, the intelligent layer is able to offload the services that are not essential in order to allow customers to finalize their purchases. Such delicate incident response strategy would also make sure the business most important functions are not compromised, even in situations where systems are partially degraded.

Lastly, the effectiveness of the framework highlights the value of a homogenous data model. With metrics, logs, and traces all being part of one smart layer, the framework circumvents the siloed data issue in which various groups consider various dashboards. The findings indicate that the decisions made by the response engine are much more correct when the entire context of a transaction is made accessible. Automated interventions have a high success rate of 95, indicating that the reasoning on how to map anomalies to actions is strong. But the 5% of cases which remain, implies that there is still a need in human supervision of the complicated situations that do not fit with the learned patterns of the structure.

7. CONCLUSION

The study has managed to illustrate the conceptualization and the execution of an Intelligent Observability and Incident Response Framework that is specific to distributed e-commerce systems. Giving up on simple monitoring and switching to the concept of deep observability, the framework presents a holistic perspective of the system health, which is granular and actionable. Current data examination of 452 data cases confirmed that automated detection and response could achieve far superior results in all main metrics, such as the detection time, repair time and the impact radius containment. The findings, which are represented in contour plots and multi-line graphs, testify to the fact that the framework successfully handles the challenges of microservices in terms of detecting early signs of failure and initiating corrective action. The tables show the efficiency of the framework in resource management and its capacity to give priority to the business-critical pathways during the periods of stress. With 95% success in automated remediation, the study demonstrates that a big percentage of operational incidents can be dealt with programmatically, and it is possible to leave the engineering workforce to higher value work instead of troubleshooting processes, which occur consistently. High-availability requirements of the current digital commerce are directly supported by the framework capability of eliminating the window of degradation of the service. Conclusively, the introduction of the intelligence into the observability stack is both an enhanced feature rather than the incremental feature to handle the scale and variability of the current distributed environments. The research offers a strong basis on which stronger, more resilient self-healing infrastructures can be developed to survive the challenges of the ever-growing technical complexity. The next generation of intelligent observability is in the change of pattern-based response to real autonomous system development. The future research that can be pursued is the combination of the generative AI models, generating the remediation scripts on the fly, used in novel incidents, which have no defined recovery procedures. At present, the framework is based on a library of known actions; a 5% gap in which manual intervention is still needed can be bridged by having dynamically generated solutions. The second component that can be explored is the application of Chaos Engineering as a continuous feedback mechanism of the smart layer. The system can learn and optimize its response strategies in a carefully monitored environment by deliberately introducing failures in the times of low traffic, continually enhancing their accuracy as predictors. Moreover, the framework may be extended to the network boundaries and even more benefits may be gained. The observability stack needs to be in line with the current transition to e-commerce where edge computing is adopted to minimize latency amongst global users. Future research might explore how the intelligent analysis layer can be distributed to ensure that the localized incidents are managed at the edge without necessarily having to go through the data center located at the center. It would also decrease the response time, as well as enhance the user experience of customers that are separated around the world. Lastly, with privacy laws being increasingly tougher, coming up with privacy-conscious observability capable of offering deep insights without having to expose sensitive customer information will be a major point of concern. The aim is to develop a system not only self-healing but naturally secure and compliant as well.

8. REFERENCES

[1] B. H. Sigelman, L. A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag, “Dapper: A Large-Scale Distributed Systems Tracing Infrastructure,” *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2010. https://research.google.com/pubs/archive/36356.pdf

[2] J. Dean and L. A. Barroso, “The Tail at Scale,” *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.https://doi.org/10.1145/2408776.2408794

[3] W. Xu, L. Huang, A. Fox, D. A. Patterson, and M. I. Jordan, “Detecting Large-Scale System Problems by Mining Console Logs,” *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, 2009.https://doi.org/10.1145/1629575.1629587

[4] J. Thalheim, A. Rodrigues, I. E. Akkus, P. Bhatotia, R. Chen, B. Viswanath, L. Jiao, and C. Fetzer, “Sieve: Actionable Insights from Monitored Metrics in Microservices,” *IEEE/ACM International Conference on Distributed Systems Platforms*, 2017.https://arxiv.org/abs/1709.06686

[5] F. Lin, K. Muzumdar, N. Laptev, M. Curelea, S. Lee, and S. Sankar, "Fast Dimensional Analysis for Root Cause Investigation in a Large-Scale Service Environment," *IEEE International Conference on Big Data*, 2019.https://arxiv.org/abs/1911.01225

[6] Y. Gan, Y. Zhang, K. Chen, et al., "Root Cause Analysis of Failures in Microservices Through Anomaly Detection," *Proceedings of the IEEE International Conference on Cloud Computing (CLOUD)*, 2019.https://ieeexplore.ieee.org/document/8812060

[7] M. Chen, A. Accardi, A. Archibald, et al., "AI for IT Operations (AIOps): Challenges and Opportunities," *IEEE Intelligent Systems*, vol. 35, no. 2, pp. 6–14, 2020.https://doi.org/10.1109/MIS.2020.2973845

[8] Z. Chen, M. R. Lyu, and Z. Zheng, "TraceMesh: Scalable and Streaming Sampling for Distributed Traces," *IEEE Transactions on Network and Service Management*, 2024.https://arxiv.org/abs/2406.06975

[9] A. Lavin and S. Ahmad, "Evaluating Real-Time Anomaly Detection Algorithms," *IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2015.https://doi.org/10.1109/ICMLA.2015.141

[10] Z. Chen et al., "An Anomaly Detection Algorithm for Microservice Architecture Based on Robust Principal Component Analysis," *IEEE Access*, vol. 8, pp. 226397–226408, 2020.https://doi.org/10.1109/access.2020.3044610

[11] Z. Chen, Z. Jiang, Y. Su, M. R. Lyu, and Z. Zheng, "TraceMesh: Scalable and Streaming Sampling for Distributed Traces," *2024 IEEE 17th International Conference on Cloud Computing (CLOUD)*, Shenzhen, China, 2024, pp. 54–65. https://doi.org/10.1109/CLOUD62652.2024.00016

[12] J. Soldani and A. Brogi, "Anomaly Detection and Failure Root Cause Analysis in Microservice-Based Cloud Applications: A Survey," *Journal of Systems and Software*, 2021.https://doi.org/10.48550/arXiv.2105.12378

[13] V.-H. Le and H. Zhang, "Log-Based Anomaly Detection Without Log Parsing," *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Melbourne, Australia, 2021, pp. 492–504.https://doi.org/10.1109/ASE51524.2021.9678773