International Journal of Computer (IJC)
ISSN 2307-4523 (Print & Online)
https://ijcjournal.org/index.php/InternationalJournalOfComputer/index

Event Correlation and Request Tracing in Asynchronous Microservice Architectures

Gleb D. Shkriabin*

Chief Technology Officer, White Code Works LLC, USA, New Jersey
Email:cto@whitecodeworks.com

Abstract

This article presents an analytical synthesis of scientific approaches to examining event correlation and request tracing in asynchronous microservice architectures. The study is conducted as a systematic analysis of peer-reviewed publications and focuses on interpreting the architectural and operational factors that determine the reconstruction of causal relationships under event-driven and hybrid inter-service communication. Particular attention is given to the impact of asynchrony, message delivery semantics, fault-tolerance mechanisms, and architectural antipatterns on the interpretability of distributed traces and the robustness of event correlation. It is demonstrated that traditional linear request tracing proves insufficient for explaining the behavior of asynchronous systems, whereas event correlation assumes the role of a foundational architectural mechanism rather than a supplementary observability tool. The findings indicate that correlation degradation exhibits a systemic nature and is largely driven by the structure of interactions among services, execution parameters, and the quality of context propagation, rather than by the selection of specific monitoring instruments. The transition toward managed analysis and response loops is shown to amplify the requirements for architectural consistency in correlation, as event-binding errors lead to distorted analytical conclusions and incorrect control interventions. The obtained results refine the architectural conditions for robust event correlation and can be applied in the design and operation of asynchronous microservice platforms with stringent demands for explainability and manageability.

Keywords: microservice architecture; asynchronous systems; event correlation; request tracing; observability; distributed traces.

Received: 2/10/2026
Accepted: 4/10/2026
Published: 4/17/2026

---

* *Corresponding author.*

1. Introduction

The evolution of microservice architectures under conditions of high load, distribution, and execution dynamics complicates diagnostics, failure analysis, and the interpretation of system behavior [3]. Transitioning to asynchronous and event-driven models enhances scalability and fault tolerance, yet it disrupts linear causal relationships between requests and events. Consequently, traditional request tracing loses its capacity to ensure end-to-end execution interpretability, making event correlation a critical prerequisite for the manageability and observability of microservice systems. Unlike tracing, which records a sequence of operations, event correlation aims to reconstruct causality among asynchronous actions that are not bound by a single call stack.

The urgency of this problem is amplified in hybrid architectures that combine synchronous calls with asynchronous message exchange. User operations within such systems fragment into independent events distributed across time and space, rendering it impossible to reconstruct a coherent execution overview without explicit context binding [8]. Repeated message delivery, alternative execution paths, and temporal desynchronization further distort causality and evade description by linear tracing. Fragmented observability, grounded in the separate analysis of requests, metrics, and logs, fails to accurately localize sources of degradation and results in a skewed interpretation of the distributed system's behavior.

The objective of this study is to perform an analytical synthesis of architectural and operational approaches to event correlation and request tracing in asynchronous microservice architectures, aiming to identify the factors that constrain the reconstruction of causal relationships. To achieve this goal, the following tasks are addressed: systematizing the architectural solutions and patterns that shape the nature of event correlation and tracing in asynchronous microservice systems; analyzing the influence of antipatterns, fault-tolerance mechanisms, and orchestration dynamics on the completeness and interpretability of distributed traces; and generalizing approaches to the analytical and machine-learning-based processing of correlated traces and logs in diagnostics and anomaly detection tasks.

The research hypothesis posits that the effectiveness of request tracing in asynchronous microservice architectures is primarily determined by the architectural consistency of event correlation mechanisms and fault-tolerance models, rather than by the specific suite of observability tools employed. The scientific novelty of this work lies in the development of an analytical framework wherein event correlation and request tracing are examined as interconnected elements of managed observability, which ensures the explainability and manageability of asynchronous microservice system behavior.

2. Materials and Methods

This study was conducted as a structured analytical review aimed at synthesizing architectural and operational factors influencing event correlation and request tracing in asynchronous microservice architectures. The methodology encompassed structured stages of publication identification, thematic grouping, and analytical evaluation, thereby ensuring consistency and transparency of the analytical process. An initial search was executed

using thematic keywords related to microservice architecture, asynchronous communication, tracing, and observability, after which sources were selected based on their relevance to the research objectives.

The final corpus was assembled from peer-reviewed scientific papers published between 2023 and 2025, dedicated to asynchronous inter-service communication, event coordination, and telemetry analysis in microservice systems. Publications providing an architectural, operational, or methodological analysis of how selected solutions impact the reconstruction of causal relationships were included. Descriptive works limited to conceptual or instrumental overviews, without addressing architectural dependencies and their consequences for event correlation and request tracing, were excluded. Each source underwent analysis according to a unified framework comprising the architectural context, the inter-service communication model, and its effect on the interpretability of distributed execution. The findings were aggregated at the level of architectural solutions, antipatterns, and analytical approaches.

The architectural causes underlying the complication of diagnostics and tracing in high-load microservice systems are analyzed by A. D. Bogutsky and his colleagues [1]. The specifics of hybrid synchronous-asynchronous architectures and the requirements for maintaining transaction integrity are examined by I. N. Nasyrova and his colleagues [2], while the limitations of request tracing at the application programming interface level in performance analysis tasks are addressed by A. Smirnov and his colleagues [3]. The influence of asynchronous interactions and message delivery semantics when utilizing message brokers is demonstrated in the study by A. E. Bazhenov and his colleagues [4].

Architectural antipatterns that degrade observability and execution interpretability are systematized by A. V. Ziborev and his colleagues [5], whereas a comparison of synchronous and asynchronous integration models for external services is presented by E. I. Lyashov and his colleagues [6]. Issues of systemic observability and the necessity of consolidating telemetry data are explored by V. Yu. Maksimov and his colleagues [7], and the infrastructural aspects of observability within service mesh layers are detailed by I. V. Myasnikov and his colleagues [8]. Approaches to the analytical processing of telemetry, including machine learning techniques, are introduced in the works of V. A. Oleinik and his colleagues [9] and D. A. Khudyakov and his colleagues [10], which propose formalized methods for correlating logs and distributed traces.

The research is analytical in nature and does not incorporate proprietary experimental measurements. The interpretation of quantitative results remains limited to the data provided within the analyzed publications. The study is conceptual and interpretative in scope and does not aim to provide statistically generalizable empirical results.

3. Results

An analysis of the reviewed publications revealed that in asynchronous microservice architectures, event correlation and request tracing emerge primarily as consequences of architectural decisions dictating the nature of inter-service communication. The shift from synchronous call chains to hybrid and event-driven interaction

models enhances system robustness and scalability, yet it simultaneously complicates the reconstruction of causal links between component actions [2]. Under these conditions, request tracing ceases to be a linear representation of execution and necessitates the architectural alignment of context propagation, message processing, and failure management mechanisms. Table 1 outlines architectural solutions and their implications for event correlation and request tracing in hybrid and asynchronous microservice architectures.

Table 1: Architectural solutions and their impact on event correlation and request tracing (Compiled by the author based on source: [2])

Architectural Solution	Impact on Correlation and Tracing	Primary Constraints
Hybrid Sync-Async Communication	Events become primary carriers of causal relationships; linear tracing loses completeness.	Absence of a formal model for merging synchronous and asynchronous chains.
Asynchronous Coordination	Correlation is constructed as a chain of domain events rather than a call stack.	Loss and distortion of causality during delays and repeated delivery.
Correlation Identifiers	Enable the logical binding of events and distributed execution segments.	Inadequately formalized transmission mechanism through message brokers.
Fault-Tolerance Mechanisms	Generate alternative execution paths and chain disruptions.	Distortion of trace interpretation without explicit recording of retries.

An analysis of the architectural solutions indicates that the inter-service communication model defines the structure and completeness of event correlation. Amidst dynamic deployment and scaling, the stability of identifiers and execution context emerges as a key factor in reconstructing causal chains [2], which shifts the role of tracing from a diagnostic tool to an element of architectural design. The deployment of asynchronous message brokers amplifies the requirements for explicitly recording causality, as variations in delivery semantics, ordering, and reprocessing lead to trace fragmentation and the need to account for alternative execution paths [4]. As a result, event correlation adopts a graph-based structure rather than a linear form.

Architectural antipatterns associated with excessive decomposition and the presence of synchronous nodes manifest as context loss and a distorted observable picture of execution [5]. Infrastructural observability mechanisms at the service mesh layer expand telemetry data collection; however, lacking an aligned architectural correlation model, they fail to ensure the interpretability of distributed traces [8]. This underscores that event correlation and request tracing must be regarded as foundational architectural mechanisms within asynchronous

microservice systems. Contemporary observability practices in distributed systems are largely shaped by widely adopted frameworks and standards, including OpenTelemetry, Jaeger, and Zipkin, as well as the W3C Trace Context specification. These solutions provide standardized mechanisms for context propagation and distributed tracing across microservices. However, their underlying models remain primarily oriented toward reconstructing linear request flows. In asynchronous and event-driven architectures, such models encounter fundamental limitations, as causal relationships are no longer confined to a single execution path. This gap motivates the need to reconsider event correlation as an architectural mechanism rather than a purely instrumental capability.

Practical analysis was directed toward examining traces as ordered behavioral chains generated within a single user request. Throughout the study, a trace was interpreted as a process comprising sequences of events and logs united by a common execution context. To reconstruct causal relationships, preliminary structuring of logs by trace identifier and span identifier was performed, which eliminated the conflation of messages from different requests and facilitated a transition from analyzing isolated records to evaluating holistic execution scenarios [10]. This approach proved indispensable in the context of asynchronous communication, where causality is not expressed through a call stack and is disrupted by queues, parallel processing, and message delivery delays. The high dynamics of microservice systems were taken into account, where fluctuations in instance counts and routing complicate the stable binding of logs to a specific request [1]. Figure 1 details the sample generation parameters utilized to construct the test traces, including the sizes of the training and control samples, the proportion of anomalous chains, trace length, the number of spans per trace, and the maximum message length.

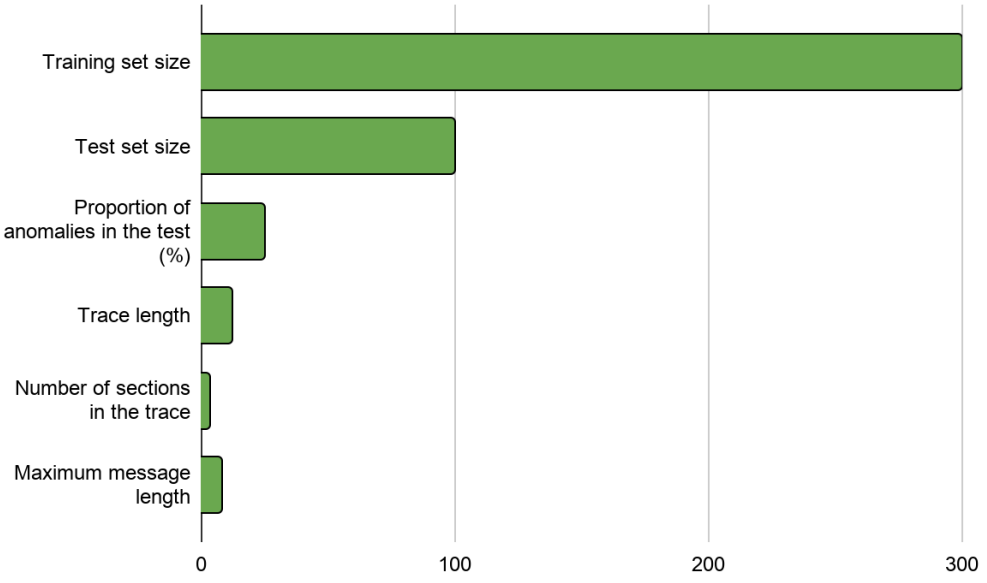

Figure 1: Parameters for sample generation used to test the event correlation prototype (Compiled by the author based on source: [10])

The following experimental configuration is adapted from the study by Khudyakov [10] and is used here solely for analytical interpretation rather than as an original empirical contribution. The numerical parameters presented

in the diagram reflect a balanced configuration of test data, tailored for analyzing event correlation under moderate structural trace complexity. According to Khudyakov [10], the training sample size of 300 traces substantially exceeds the test sample size of 100 traces, securing robust training and validation of the correlation mechanisms. The 25% proportion of anomalous chains establishes relatively rigorous testing conditions wherein every fourth trace exhibits deviations from normal behavior. An average trace length of 12 events and a limit of 3 spans per trace indicate the analysis of typical user scenarios devoid of excessive decomposition depth. A maximum message length, capped at 8 arbitrary units, further curtails log variability and allows the focus to remain on reconstructing causal relationships rather than processing superfluous textual noise. The present study does not perform independent experimental validation and instead focuses on interpreting the architectural implications of the reported results.

Trace length and the number of spans within it served as parameters for the depth of distributed execution. Escalating these values simulated architectures with a high degree of decomposition and numerous inter-service transitions, where the risk of context loss and the emergence of ambiguous event linkages intensifies [5]. The specifics of asynchronous exchange via message brokers were considered, where repeated delivery and reordering generate alternative chain branches, necessitating a distinction between delivery retries and action retries at the analytical level. Restricting the maximum message length was employed to align the test data more closely with operational logs and to elevate the requirements for their parsing and normalization.

The generated behavioral chains were utilized as a foundation for analyzing event correlation and trace interpretability. The accuracy of preliminary log binding directly governed the stability of the identified sequences and reduced the analytical sensitivity to noise and trivial record variations, aligning with the practices of telemetry analytics in microservice systems.

4. Discussion

An analysis of the results demonstrates that the degradation of event correlation and trace interpretability in asynchronous microservice systems is systemic in nature and cannot be reduced to isolated implementation errors or deficiencies in observability tools. A pivotal role in shaping these distortions is played by architectural antipatterns, which embed structural constraints on the reconstruction of causal relationships at the design phase itself. Within asynchronous architectures, it is the orchestration of interactions between services that determines whether cohesion among component actions is preserved or if causality collapses within the telemetry data. In this context, antipatterns should be viewed as architectural sources of observability degradation, as they disrupt context propagation, blur transaction boundaries, and warp the structure of behavioral chains, rendering tracing incapable of reflecting the system's actual execution [5]. Table 2 illustrates how key microservice architecture antipatterns impact event correlation and request tracing.

Table 2: Microservice antipatterns and their impact on correlation and tracing (Compiled by the author based on source: [5])

Antipattern	Primary Architectural Problem	Effect on Event Correlation	Effect on Request Tracing
Excessive Microservice Granularity	Over-decomposition and a surge in network interactions ("Nano-services").	Loss or duplication of context between events.	Elongated, fragmented, and difficult-to-interpret traces.
Synchronous Nodes	Blocking call chains within an otherwise async system.	Blurring of causality across the entire execution chain.	Inability to localize delay sources without a complete, uninterrupted trace.
Shared Databases	Hidden dependencies through a common repository (Database-as-IPC).	Events fail to reflect the true causes of state changes.	Tracing fails to capture peripheral write operations outside the service logic.
Neglect of Monitoring	Absence of systemic telemetry and standardized headers.	Impossibility of reconstructing event chains after the fact.	Total loss of end-to-end visibility over request execution.

Examining the antipatterns reveals that excessive decomposition violates the proportionality between a system's architectural complexity and the feasibility of robust correlation. With an abundance of granular services, the execution context splinters faster than tracing tools can capture it, leading to the fragmentation of behavioral chains and an increase in ambiguous linkages between events. Synchronous nodes engender the opposite problem. They centralize control yet render causality opaque, as delays and failures propagate across the entire chain without an explicit reflection in the event structure. The utilization of shared databases generates latent causal links that do not appear in telemetry and cannot be reconstructed via message correlation. The absence of systemic monitoring exacerbates this effect, as even potentially recoverable connections remain unrecorded and inaccessible for analysis.

Architectural antipatterns do not exhaust the factors contributing to the degradation of event correlation and trace interpretability. Even within a formally sound architecture, execution behavioral characteristics, trace generation parameters, and log structuring methodologies play a substantial role. This shifts the analytical focus from static architectural decisions to dynamic execution chains, within which correlation is formed and distorted at the level of specific request processing scenarios.

The evolution of observability in asynchronous microservice architectures manifests as a transition from passive monitoring to managed analysis and response loops [7]. Under conditions of high distribution, simply logging metrics and journals ceases to provide explainability for system behavior. Event correlation assumes paramount importance, enabling the binding of disparate signals into coherent behavioral chains and utilizing them as a foundation for analysis and managerial interventions.

Within such loops, event correlation acts not as an auxiliary visualization tool, but as a structural mechanism dictating the quality of analytical conclusions. Consolidating metrics, logs, and traces into a unified execution context facilitates the shift from logging symptoms to pinpointing sources of degradation, with data consistency and causal linkage becoming the decisive factors rather than data volume. Absent robust correlation, automated analysis leans on fragmented information and replicates the errors of conventional monitoring.

Advancements in telemetry analytics intensify these requirements. Anomaly detection and forecasting algorithms operate on event sequences, meaning the robustness of their results relies entirely on the accurate reconstruction of execution chains. In asynchronous architectures, this necessitates accounting for duplicate message delivery, parallel processing, and mutable execution topologies. The formation of closed analysis and response loops heightens the importance of architectural consistency in observability, given that correlation errors yield distorted conclusions and improper control actions.

Deploying distributed tracing in conjunction with analytical models allows the behavior of an asynchronous microservice system to be viewed as a dynamic process rather than a collection of isolated incidents. Nevertheless, such a shift exposes a fundamental tension between architectural dynamism and interpretation stability: as asynchrony, parallelism, and management automation escalate, the sensitivity of analytical loops to event correlation errors heightens. Under these conditions, the interpretability of system behavior is determined less by the sophistication of the applied analytical methods and more by the architectural constraints on reconstructing causal links. This challenges the universal applicability of existing approaches to managed analysis and response loops and demands a reassessment of event correlation's role as a fundamental prerequisite for the stable and explainable operation of asynchronous microservice architectures.

5. Conclusion

The research demonstrates that within asynchronous microservice architectures, event correlation and request tracing shed their supplementary nature and evolve into foundational architectural mechanisms that govern the explainability and manageability of system behavior. Asynchrony, execution parallelism, and event coordination dismantle the linear execution model, making it impossible to reconstruct causal relationships without explicit and synchronized context management. The findings further indicate that no universal request tracing model can ensure causal interpretability in asynchronous microservice architectures, as tracing effectiveness is inherently constrained by the architectural consistency of event correlation mechanisms rather than by the capabilities of observability tools. Under these conditions, a trace is no longer a simple sequence of calls and must be treated as a behavioral chain encompassing alternative paths, delays, and repetitive actions.

It is established that correlation degradation is systemic in nature and largely dictated by architectural antipatterns. Excessive decomposition, synchronous nodes, hidden dependencies via shared repositories, and a lack of systemic telemetry distort the structure of behavioral chains and obscure causality. Even with formally precise tracing, such architectural choices result in context loss, trace fragmentation, and an increase in ambiguous linkages among events. This corroborates that the interpretability of distributed execution is defined not by the volume of gathered data, but by the architectural provisions for binding it.

The transition from passive monitoring to managed analysis and response loops amplifies the significance of robust event correlation. Within such loops, binding errors lead to a distorted interpretation of system behavior and flawed managerial interventions. This mandates viewing event correlation as an architectural prerequisite for the stable operation of asynchronous microservice systems and as a foundation for constructing analytical and control mechanisms capable of functioning amidst the high dynamics and uncertainty of distributed execution. The practical significance of these findings lies in the potential to apply the proposed analytical framework when designing observability architectures and event correlation strategies in industrial asynchronous microservice systems.

References

[1]. Bogutskii, A. (2025). Design and implementation of a microservices architecture in high-load distributed systems. Universum: Technical Sciences, 11(140), 21–25. Available at: https://cyberleninka.ru/article/n/design-and-implementation-of-a-microservices-architecture-n-high-load-distributed-systems (accessed January 21, 2026).

[2]. Nasyrova, I. N. (2025). Microservice architectures for financial platforms: Challenges and solutions. Professional Bulletin: Information Technology and Security, 2, 49–55. Available at: https://cyberleninka.ru/article/n/microservice-architectures-for-financial-platforms-challenges-and-solutions (accessed January 21, 2026).

[3]. Smirnov, A. (2025). Methods for detecting and resolving issues in APIs: Profiling and performance optimization. Cold Science, 14, 7–15. Available at: https://cyberleninka.ru/article/n/methods-for-detecting-and-resolving-issues-in-api-profiling-and-performance-optimization (accessed January 22, 2026).

[4]. Bazhenov, A. E., Raikov, A. V., Nekhaev, M. V., & Orekhovsky, N. V. (2025). Optimization of microservice interaction using asynchronous calls and message brokers (Kafka, RabbitMQ). Software Systems and Computational Methods, 4, 77–93. Available at: https://cyberleninka.ru/article/n/optimizatsiya-vzaimodeystviya-mikroservisov-s-ispolzovaniem-asinhronnyh-vyzovov-i-brokerov-soobscheniy-kafka-rabbitmq (accessed January 22, 2026).

[5]. Ziborev, A. V. (2023). Antipatterns in building microservice applications in high-load projects. Universum: Technical Sciences, 11-1(116), 29–34. Available at: https://cyberleninka.ru/article/n/antipatterny-postroeniya-mikroservisnyh-prilozheniy-v-vysokonagruzhennyh-proektah (accessed January 23, 2026).

[6]. Lyashov, E. I. (2025). Integration of external services into distributed applications based on Spring.

Innovative Science, 5-1-1, 67–73. Available at: https://cyberleninka.ru/article/n/integratsiya-vneshnih-servisov-v-raspredyonnye-prilozheniya-na-baze-spring (accessed January 23, 2026).

[7]. Maksimov, V. Yu. (2024). Overcoming observability challenges in microservice architectures. Innovative Science, 2-1, 30–35. Available at: https://cyberleninka.ru/article/n/preodolenie-trudnostey-nablyudaemosti-v-mikroservisnoy-arhitekture (accessed January 24, 2026).

[8]. Myasnikov, I. V. (2025). Monitoring Istio components to ensure reliability and observability of a service mesh. Bulletin of Science, 5(86), 734–744. Available at: https://cyberleninka.ru/article/n/monitoring-komponentov-istio-dlya-obespecheniya-nadezhnosti-i-nablyudaemosti-service-mesh (accessed January 24, 2026).

[9]. Oleinik, V. A., & Kartbaev, A. Zh. (2024). Integration of machine learning methods into monitoring and analysis systems for microservices. Universum: Technical Sciences, 12(129), 62–68. Available at: https://cyberleninka.ru/article/n/integratsiya-metodov-mashinnogo-obucheniya-v-sistemu-monitoringa-i-analiza-mikroservisov (accessed January 25, 2026).

[10]. Khudyakov, D. A. (2023). Development of an anomaly detection system based on distributed log tracing. Vestnik of Novosibirsk State University. Series: Information Technologies, 1, 62–72. Available at: https://cyberleninka.ru/article/n/razrabotka-sistemy-vyyavleniya-anomaliy-na-osnove-raspredelennoy-trassirovki-logov (accessed January 25, 2026).