RESEARCH & INNOVATION DIVISION
IJIST

# FinOps–AIOps Fusion: Cost-Aware Anomaly Attribution for Microservice-Based Cloud Systems

Muhammad Jazib, Muhammad Haseeb Anees
Department of Computer Science, University of Central Punjab, Lahore, Pakistan
*Correspondence: muhammadjazib471@gmail.com, haseebanees832@gmail.com

**Citation |** Jazib. M, Anees. M. H, "FinOps–AIOps Fusion: Cost-Aware Anomaly Attribution for Microservice-Based Cloud Systems", IJIST, Special Issue pp 418-446, May 2026

**Received |** March 20, 2025 **Revised |** April 30, 2026 **Accepted |** May 07, 2026 **Published |** May 11, 2026.

Cloud-native microservice systems produce so much operational telemetry and cloud billing data that the task of maintaining a balance between system reliability and cost efficiency is growing more difficult. The current Artificial Intelligence-based IT Operations (AIOps) software is based on identifying technical anomalies, and Financial Operations (FinOps) software has cost visibility without associating costs with operational events. In this paper, a combined FinOps-AIOps hybrid model of cost-conscious anomaly attribution at the microservice level is proposed. The framework combines data from distributed tracing, system monitoring, and cloud billing in a fusion engine where the identified anomalies are correlated with their financial effect. The proposed system was tested within a simulated microservice environment based on Kubernetes with 20 services and a 7-day workload. The results of the experiments show that the framework achieves 92% anomaly detection accuracy with a precision of 0.90, a recall of 0.88, and an F1-score of 0.89. Also, the model can attribute costs with about 85% accuracy, and cost-aware incident prioritization is improved by 62% compared to AIOps-only models. The results show a statistically significant improvement in cost optimization efficiency without degrading detection performance (F1 deviation < 3%). The results suggest that operational intelligence combined with financial analytics can support real-time, cost-sensitive decision-making in real-time and cost-sensitive to enhance the reliability of the system and cost optimization in the cloud setting of distributed environments.

**Keywords:** Distributed Tracing, Anomaly Detection, Cloud Native Systems, FinOps, and AIOps

**Introduction:**

The rise of cloud-native microservice architecture has revolutionized the development, deployment, and maintenance of modern applications. As opposed to developing monolithic systems, enterprises are now based on loosely coupled microservices that communicate using APIs and asynchronous message queues. This paradigm shift improves scalability, fault tolerance, and development agility; however, it also introduces significant challenges in operational monitoring and financial management. Each microservice operates within its own containerized environment, generating distributed traces, logs, and performance metrics for heterogeneous infrastructure levels and multiple cloud providers. As these systems scale, manual monitoring becomes infeasible, which led to the adoption of Artificial Intelligence for IT Operations (AIOps). AIOps is enabled through data-driven methodologies and machine learning techniques to automate the detection of anomalies, predictions of incidents, and root cause analysis. It is aimed at reducing the cognitive load on these operators and improving system reliability by learning normal behavioral patterns and flagging deviations in real time. Despite these advancements, AIOps tools remain limited to technical anomaly detection, such as latency spikes, another name for API timeouts, and throughput degradation. However, they do not translate these events into monetary terms that can guide decision makers. Recent advancements in AIOps, including the use of large language models, have significantly improved anomaly detection and root cause analysis in cloud-native systems [1]. For instance, an anomaly in a payment processing microservice may cause significant overconsumption of cloud resource consumption, resulting in hidden costs, not immediately visible to the FinOps (Financial Operations) teams. In parallel, FinOps has emerged of FinOps as a cultural and technical practice in cloud spending control. FinOps focuses on cost transparency, accountability, and optimization by promoting collaboration between financial, engineering, and business teams. Organizations use FinOps to track the costs of cloud expenditure, enforce budgeting policies, and allocate costs across services or departments [2][3]. However, FinOps systems are usually reactive; they track costs only after they have occurred, and they do not have the operational awareness to grasp why a certain spike in cost happened. This gap translates into delayed reaction times, misaligned budgets, and little visibility into the root causes of inefficiencies in costs. Despite independent advancement in both fields, there is at present no holistic mechanism for the correlating operational anomalies detected by AIOps, along with their financial implications, taken care of by FinOps. Existing AIOps platforms offer root cause analytics and scores of anomalies, but fail to quantify the monetary cost of the issue detected [4][5]. On the other hand, FinOps tools can analyze cost data but cannot link expenses to particular operational anomalies or service level degradations. This disconnect causes organizations to understand which microservices are both operationally inefficient and financially expensive, thereby limiting optimization potential. To overcome this limitation, this research presents the unified FinOps AIOps Fusion Framework to integrate the operational intelligence of AIOps with the cost governance principles of FinOps. The framework is aimed at providing cost Awareness in Anomaly attribution by meaning every anomaly detected in the system is mapped to its financial footprint between microservices, containers, and cloud assets. This approach enables teams not only to detect what went wrong but also to know how much it costs and which service or infrastructure part is responsible. The motivation for this fusion is the changing nature of cloud economics & system observability. As multicloud environments become more prevalent, cost attribution becomes more and more complex due to dynamic scaling, resource pooling, and x-cloud billing mechanisms. AIOps offers how granular and automated the data should be to track these dynamic behaviors, and FinOps gives the accountability scheme for converting technical anomalies to financial outcomes. By combining the two, organizations can achieve a constant feedback loop between the performance of the system and cost efficiency. This study aims to achieve three objectives:

To create a general architecture that links invoicing and expense records with distributed tracing data.

To provide an algorithmic methodology that uses resource usage to assign fees associated with anomalies to specific microservices.

To show via experimental simulation that cost-sensitive anomaly attribution lowers needless cloud spending and improves operational prioritizing.

The study has a systematic workflow aimed at combining operational analytics and financial cost management on cloud native platforms. First, observability tools are used to collect telemetry and distributed tracing data in the microservice-based cloud environment. These operational indicators are then fed into anomaly detection systems in the AIOps layer to detect abnormal behavior in the system. FinOps analytics are used in the next step to link the cloud billing and resource utilization data with the financial quantification of the identified anomalies. The results of the two layers are fused in a fusion engine, which undertakes cost-sensitive anomaly attribution at the microservice level. Lastly, the framework is tested using experimental analysis to determine the performance of anomaly detection and how it can be used to provide cost-aware operations decisions.

It seeks to fill the gap between operational anomaly detection and financial cost management in cloud native microservice environments. The purpose of the given work is to create and test a single framework that involves Artificial Intelligence to operate IT (AIOps) and Financial Operations (FinOps), and allows cost-sensitive anomaly detection and attribution. This piece of work fills the gaps in technology and economy of the detection of anomalies and contributes to the existing literature in the field of intelligent cloud management. The innovation of this study is the association of real-time AIOps indicators with FinOps cost analytics to transform FinOps into a data-driven decision support system that is not a reactive monitoring strategy. Besides offering a baseline of sustainable cloud economics, the given methodology will contribute to a higher level of operational applicability of AIOps research, as it will allow a company to categorize operational anomalies in terms of financial significance.

The last developments in cloud observability and anomaly detection have been studied intensively in previous literature [4][5][6]. Still, the vast majority of the current solutions are either performance tracking or failure identification, and do not incorporate cost-consciousness into operational intelligence. Moreover, the methods of optimization, based on FinOps, are more focused on cost distribution and management [7] without adaptation to anomalies in real-time. This is where a common FinOps-AIOps framework is required.

The recent progress of cloud computing and microservices architectures has made the monitoring of operations and cost management a lot more complicated. Recent research 2022-2025 focuses on the increasing relevance of combining Artificial Intelligence to IT Operations (AIOps) with financial analytics in order to provide smart and cost-efficient cloud management. Research is emerging on the application of machine learning, distributed tracing, and multimodal observability as tools to detect anomalies in microservices, and FinOps has changed to facilitate real-time cost visibility and shared financial responsibility among engineering teams. Nevertheless, even with these innovations, the current solutions are mostly siloed, as AIOps is concerned with the performance anomalies and FinOps with the optimization of costs separately. This division does not allow organizations to learn the financial consequences of operational inefficiencies in real time.

**Problem Statement:**

In spite of the considerable progress made in both AIOps and FinOps, there is still a serious void in the possibility of correlating operational anomalies with their financial impact in real time. Current AIOps tools are used to monitor the presence of a performance degradation, e.g., latency spikes, resource inefficiencies, but not to measure the cost

consequences. On the other hand, FinOps platforms offer cost visibility and cost optimization insights, but do not have the ability to determine the operational causes of cost changes. This separation inhibits the power of organizations to prioritize incidents in terms of technical severity and financial effects. Thus, an integrated framework capable of real-time, microservice-level cost-conscious anomaly attribution is necessary to support decision-making in cloud-native environments.

**Novelty:**

The study is novel in the following sense of the word. The proposed study involves several original contributions that make it stand out among the current literature in the areas of AIOps and FinOps:

The fusion framework is a single architecture combining both operational telemetry and financial cost analytics, with a common FinOps-AIOps framework.

An anomaly attribution engine that is cost-aware in real-time and able to map anomalies that are detected to their financial effect on a microservice level.

A cost-weighted anomaly scoring model that ranks incidents by both technical and economic importance.

Multimodal data fusion: A method to use distributed tracing, system metrics, and cloud billing data to make better decisions.

Experimental validation: An experimental validation showing a better cost optimization and prioritization efficiency than traditional AIOps-only and FinOps-only solutions.

All these contributions will help to fill the current gap between operational intelligence and financial accountability in cloud native systems.

**Research Objectives:**

The main aim of the study is to come up with and test a single FinOps-AIOps framework of cost-sensitive anomaly attribution in cloud-native microservice settings. To accomplish this, the following measurable objectives are defined in the study:

To develop an integrated architecture to integrate AIOps telemetry data with FinOps billing data to analyze them in real-time.

To demonstrate that distributed trace-based learning models can detect anomalies with an F1-score of at least 0.85.

To develop a cost attribution system capable of achieving at least 80–85% accuracy in the cost mapping of anomalies to the appropriate microservices.

To achieve at least a 50% improvement in cost-effective incident prioritization efficiency relative to more traditional AIOps-only solutions.

To confirm the performance of the suggested framework by means of experimental simulation with quantitative measures, including precision, recall, attribution accuracy, and cost savings.

These goals will make sure that the proposed framework is not only technically useful but also practical in real-life cloud.

**Paper Organization:**

The remainder of this paper is organized as follows. Section II presents the background concepts related to AIOps, FinOps, and Distributed tracing records the entire execution pathways. Section III reviews existing literature and highlights the research gap in cost-aware anomaly detection. Section IV introduces the proposed FinOps–AIOps Fusion Framework and describes its architectural components. Section V explains the methodology and experimental design used to evaluate the framework. Section VI discusses the results and practical implications of the proposed approach. Finally, Section VII concludes the paper and outlines directions for future research.

**Background Information and Theoretical Base:**

The de facto infrastructure for delivering scalable, elastic, and globally dispersed applications is now cloud computing. Microservice architecture is becoming more and more popular. In which a large application is broken down into smaller, separately deployable services. Every microservice contains a particular business feature and interacts with other microservices via well-defined interfaces, usually message queues or REST APIs. This modular strategy generates a highly dynamic operational environment with complex interdependencies, albeit improving scalability and maintainability. Such systems demand sophisticated analytical techniques that go beyond conventional rule-based methods for monitoring, managing, and optimization.

**Artificial Intelligence for IT Operations (AIOps):**

Gartner developed the research and industry paradigm known as AIOps, which describes the use of statistical models, machine learning, and artificial intelligence to automate IT operations changes. By utilizing the massive amounts of telemetry produced by contemporary software systems, including logs, traces, metrics, and events, AIOps seeks to enhance problem identification, root cause investigation, and system remediation. AIOps technologies enable proactive anomaly detection by continuously learning the typical operational behavior of apps and infrastructure components rather than depending on static thresholds or manual dashboards. Data ingestion, correlation-based noise reduction, anomaly detection, root cause inference, and automated remediation are some of the functional steps that AIOps usually entails.

Techniques like self-supervised learning, deep neural networks, and unsupervised clustering are employed to identify minute deviations in service behavior [8][9]. [8], for example, presented a self-supervised anomaly identification method using dispersed traces that finds anomalous patterns without the need for labeled training data. In a similar vein, [9] achieved great accuracy in microservice contexts by creating deep Bayesian networks for service-level trace anomaly detection. Multilayered fault detection employing transformers [10], semi-supervised learning techniques [6], and natural language analysis are recent developments in anomaly detection for microservices, processing methods used in data tracing [11]. These techniques show that, in comparison to single modality approaches, combining various data sources—traces, logs, and metrics—produces more robust detection [12][13]. The majority of AIOps systems, however, continue to be operationally focused; they spot anomalies in performance or dependability but are not financially conscious. Therefore, even though

Although they are able to identify performance degradation, they are unable to estimate the financial consequences of excessive resource utilization, needless scaling, or cascading failures. The necessity of integrating AIOps outputs with cost-conscious systems is driven by this constraint. It spots anomalies in performance or dependability, but is not financially conscious.

Although they are able to identify performance degradation, they are unable to estimate the financial consequences of excessive resource utilization, needless scaling, or cascading failures. The necessity of integrating AIOps outputs with cost-conscious systems is driven by this constraint.

**Financial Operations (FinOps):**

FinOps, or Cloud Financial Operations, is a management discipline that combines engineering decision-making with financial accountability to maximize cloud spending, building [14]. According to the FinOps Foundation, it is a process that unites engineering, business, and finance teams to work together on cloud cost optimization using automated visibility and common KPIs. FinOps functions in a constant feedback loop of monitoring, evaluating, and optimizing cloud consumption, in contrast to traditional cost management.

Aligning spending with business value is a fundamental tenet of FinOps. Organizations may make wise trade decisions by monitoring which microservices or workloads drive cloud expenses.

FinOps enables trade-offs between spending and performance. AI and automation are used by advanced FinOps systems to expedite tagging, budgeting, and billing pattern anomaly detection. For instance, [2] presented a multi-cloud cost allocation framework powered by AI that automates financial operations in a variety of cloud environments. ABACUS, a FinOps service specifically created for cloud cost optimization through automated resource management, was also introduced by [3]. FinOps procedures encounter difficulties because of the inherent intricacy of multi-cloud settings and fluctuating costs concepts, and resources for shared infrastructure [15]. Correctly assigning expenses to particular departments, teams, or services demands thorough insight into how resources are used, frequently lacking in traditional monitoring systems. Therefore, financial anomalies, like unexpected cost increases or inefficiencies brought on by system mistakes, are discovered only after they have occurred. Considerable financial harm. Including AIOps insights in Pipelines for FinOps could change this paradigm from reactive to proactive control of expenses.

**Distributed Tracing and Observability:**

The foundation of contemporary observability is distributed tracing [16] records requests' entire execution pathways as they pass through several microservices, allowing developers and operators to comprehend dependencies, causal links, and performance bottlenecks. Together with metadata like timestamps, latency, resource IDs, and contextual tags, each trace is made up of several spans that represent actions within services. The usefulness of distributed tracing for anomaly identification and root cause analysis has been shown by recent studies. OpenTracing-based methods can successfully identify anomalies in cloud-based microservice systems, as demonstrated by [17]. MicroRCA and MicroDiag are two frameworks created by [18][19] that use distributed traces for microservice architecture performance diagnosis and root cause localization. Trace data offers a strong basis for connecting operational irregularities to financial repercussions when paired with resource use and billing information. For example higher resource consumption and expenses can be directly linked to an unusual rise in latency or CPU usage in a specific service. Using trace and log data, several studies have investigated root cause localization [9][20] showing that causal inference methods can increase diagnostic precision. Distributed traces act as the common link between AIOps and FinOps in the context of this study. These traces are processed by the FinOps subsystem to match observed events with billing data, and by the AIOps subsystem to identify anomalies. We are able to connect performance abnormalities with cost fluctuations in real time by utilizing resource tags and trace identifiers.

**Theoretical Integration of AIOps and FinOps:**

The suggested FinOps AIOps fusion is theoretically based on explainable machine learning and multimodal data fusion. Heterogeneous integration is referred to as multimodal fusion. Diverse data sources into a single analytical framework, including logs, traces, metrics, and billing data. The system can now measure the business effect of abnormalities in addition to identifying them, thanks to this integration. For instance, a sharp rise in database read latency might be linked to increased storage I/O expenses, producing a quantifiable financial signal. Explainability is also essential in the FinOps and AIOps areas. While FinOps systems must offer interpretable anomaly scores that operators may rely on, AIOps models need accountable cost allocation models that are visible. To guarantee traceability from anomaly detection to cost attribution, the fusion architecture uses explainable models like interpretable regression and attention-based neural networks. In conclusion, the convergence of three disciplines forms the theoretical basis of this study: financial attribution via FinOps, anomaly detection via AIOps, and the linkage of data via distributed tracing. These ideas work together

to create a paradigm for cost-conscious anomaly attribution that might assist data-driven decision-making in contemporary cloud systems. The conceptual differences between AIOps-only, FinOps-only, and integrated approaches are illustrated in Figure 1.

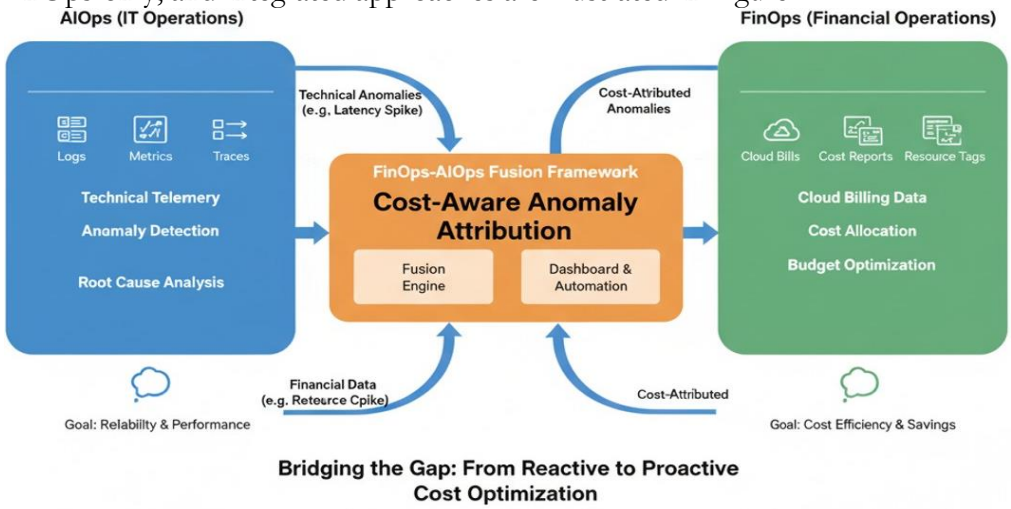


**Figure 1.** Conceptual comparison of cloud cost behavior under different operational strategies (AIOps only, FinOps-only, and integrated FinOps–AIOps). The figure illustrates how isolated operational or financial monitoring can leave cost inefficiencies unresolved, whereas integrated anomaly-cost correlation enables earlier intervention and lower cumulative cost growth. This visual motivates the need for a unified cost-aware anomaly attribution framework.

**Related Work:**

Over the past five years, research on anomaly detection, cost attribution, and operational intelligence in cloud native systems has advanced significantly. Here is a summary of the most pertinent contributions in three areas:

Anomaly detection based on AIOps

Automation of FinOps and cost optimization

Integrated frameworks that combine economics and observability

A comprehensive comparison reveals how current methods are unable to provide microservice environments with end-to-end, cost-aware anomaly attribution.

**AIOps-Based Anomaly Detection in Microservices:**

The main goal of AIOps for microservice architectures is to identify performance and reliability problems from a variety of observability data. [8] Presented a self-supervised anomaly detection method that does not require labeled data and learns behavioral patterns from scattered traces. Their approach successfully detects operational anomalies in production settings. A thorough survey of anomaly detection and failure root cause analysis in microservice-based cloud applications was given by [4][14]. Highlights of their job include the significance of integrating distributed traces with profiling metrics like CPU and memory usage to enhance comprehension of microservice degradations at the fine-grained level. Transformer topologies for multilayered fault detection, semi-supervised learning methods, and graph-based deep learning strategies that integrate trace and log data are examples of recent developments. Deep Bayesian networks have been shown by [9] to be highly accurate in detecting trace anomalies at the service level. Root cause analysis has been the specific subject of several investigations. MicroRCA, a framework for identifying microservices performance problems via causal analysis, was introduced by [18], based on the evaluation of dispersed traces. Methods for locating root cause metrics using log anomaly detection were developed by [9]. AutoMAP, which uses hierarchical causation graphs to automatically diagnose microservice-based online systems, was introduced by [13]. Robust multi-modal approaches

[12], NLP-based tracing data processing [11], and OpenTracing-based detection systems [17] have all been the subject of more recent research. CausalRCA, which employs causal inference for accurate fine-grained root cause localization, was introduced by [15]. Eadro, an end-to-end troubleshooting platform that integrates several data sources, was created by [21]. These systems are operationally limited even if they provide precise detection; abnormalities are reported as technical occurrences rather than quantifiable economic hazards. Intelligent observability systems further enhance anomaly detection by integrating multiple data modalities [22]. Not one of them evaluates the financial consequences of protracted anomalies, such as higher compute costs, overprovisioning, or SLA fines. However, these approaches focus primarily on technical anomalies and do not consider cost implications. Emerging approaches leveraging large language models further enhance anomaly detection capabilities in complex distributed systems [23].

**Advances in FinOps and Cloud Cost Optimization:**

FinOps has become a formal subject that deals with cloud computing's financial management in tandem with AIOps research. [2] Demonstrated an AI-powered multi-cloud cost allocation system that uses automation to change FinOps. The fundamental FinOps concepts of openness and shared accountability among cloud providers are embodied in this architecture. ABACUS, a FinOps service created especially for cloud cost optimization, was introduced by [3]. Recent studies emphasize AI-driven techniques for dynamic cloud cost optimization and efficient resource allocation in distributed environments [22]. Based on past consumption trends, the system offers automatic cost analysis and recommendations for resource allocation. [15] Examined the difficulties in controlling cost and scalability in microservices architecture, emphasizing the conflict between limitless scalability and budgetary restrictions. Advanced resource management has also been used to investigate automation within FinOps [16]. RAMBO is a method developed that employs Bayesian optimization for source distribution in microservices, exhibiting quantifiable increases in cost effectiveness. However, these methods do not correlate with technical anomalies found by AIOps systems and only rely on operational or financial telemetry. Financial attribution and budgeting analytics are two areas where FinOps literature generally shines, but it functions mainly independently of operational telemetry pipelines. Because of this, FinOps tools are unable to detect the financial waste that results from anomalies like a runaway container loop or an ineffective database query until after the billing cycle. Several architectural patterns for cost-efficient cloud computing have also been identified in recent literature [24]. AI-driven cost optimization has become increasingly important for managing cloud expenditure efficiently [22]. Energy-efficient and cost-aware resource management strategies have also been proposed to optimize cloud infrastructure utilization [25]. These methods improve cost visibility but lack integration with real-time operational anomalies.

**Efforts toward Integrated Frameworks:**

Notaro et al.'s recent paper [5] offered a thorough analysis of AIOps approaches for failure management, emphasizing that the main research problem is combining unified frameworks for diverse telemetry, including billing data, logs, metrics, and traces. Micro Diag was created by [18] for fine-grained performance diagnostics in microservice systems; it mostly concentrates on technical indicators without financial integration. The relationship between operational excellence and financial accountability has not received as much attention as the integration of DevOps approaches with microservices [6]. Using temporal and spatial data analysis, [8] suggested an intelligent anomaly detection technique that showed increased detection accuracy but lacked cost attribution capabilities. To date, no system has provided an efficient, real-time method to quantify anomaly-induced financial impact down to the level of microservices. Temporal alignment, uneven tagging, and the enormous cardinality of cloud

resource identifiers are the primary technological obstacles. Existing studies do not provide a unified framework for real-time anomaly-cost attribution at the microservice level.

**Table 1.** Comparative benchmark of representative AIOps, FinOps, and integrated approaches across five capabilities: anomaly detection (Anom.), profiling Prof, root cause analysis (RCA), cost attribution (Cost), and cross-domain fusion (Fusion). The table highlights that prior studies typically address only a subset of these capabilities, motivating the need for a unified framework that jointly supports real-time anomaly detection and microservice-level financial attribution.

| Study | Anom. | Prof. | RCA | Cost | Fusion |
|---|---|---|---|---|---|
| [3] | ✓ | | ✓ | | |
| [11] | ✓ | ✓ | | | |
| [10] | ✓ | | | | |
| [26] | ✓ | ✓ | ✓ | | |
| [6] | ✓ | | ✓ | | |
| [19] | | ✓ | ✓ | | |
| [14] | ✓ | ✓ | ✓ | | |
| [2] | | | | ✓ | |
| [17] | | | | ✓ | |
| [8] | ✓ | | | ✓ | Partial |
| [21] | | | | ✓ | |
| [27] | ✓ | ✓ | ✓ | | |

**Comparative Analysis and Research Gap:**

AIOps, FinOps, and integrated methods representative studies are compiled in Table I. Every task is assessed based on five competencies:

Identifying anomalies
Profiling metrics
Real-time fusion
Cost allocation
Root cause localization

Table 1 substantiates the central research gap addressed in this paper. Although prior studies demonstrate strong capabilities in anomaly detection, profiling, or cost optimization individually, none of the reviewed approaches simultaneously provide fine-grained anomaly detection, root cause localization, cost attribution, and cross-domain fusion at the microservice level. This comparison directly motivates the proposed FinOps–AIOps fusion framework, which is designed to unify these otherwise fragmented capabilities within a single operational and financial decision pipeline.

No strategy fully addresses every aspect at once, according to the comparative analysis. As can be shown, none of the assessed studies integrate microservice-level financial attribution, root cause localization, and detailed anomaly detection into a single design. Most either only concentrate on cost analytics or stop at anomaly classification. This dispersion highlights the fundamental research gap that our study attempts to fill: the lack of a comprehensive framework that can directly quantify the economic impact of technical issues and perform cost-aware anomaly attribution.

**Summary of Insights:**

Three key conclusions may be drawn from the foregoing review:

**Cross-domain fusion is necessary:** Current FinOps and AIOps systems function independently, which results in missed chances for anomaly prioritizing based on cost.

**Absence of Granular Attribution:** Instead of operating at the microservice or span level, where anomalies start, current cost allocation systems operate at the service or cluster level.
**Gap in Real Time practicality:** Although post hoc financial analysis is covered in a number of studies, few analyze how cost data might influence operational decisions in real time.

These observations serve as the foundation for our suggested FinOps–AIOps fusion architecture, which is explained in the section that follows. It specifically focuses on actionable prioritizing mechanisms, explainable cost attribution, and real-time trace to cost correlation.

A number of new solutions have tried to enhance the anomaly detection methods and root cause analysis of cloud-native systems. Indicatively, authors like use machine learning and distributed tracing to conduct automated fault diagnosis, whereas authors like concentrate on representation learning to detect anomalies. Such approaches, though effective, do not explicitly implement financial optimization, thus limiting their use in cost-sensitive cloud environments. This weakness is the reason why it is proposed to combine FinOps and AIOps.

**Proposed Finops–Aiops Fusion Framework:**

The body of research shows that operational anomaly detection (AIOps) and financial

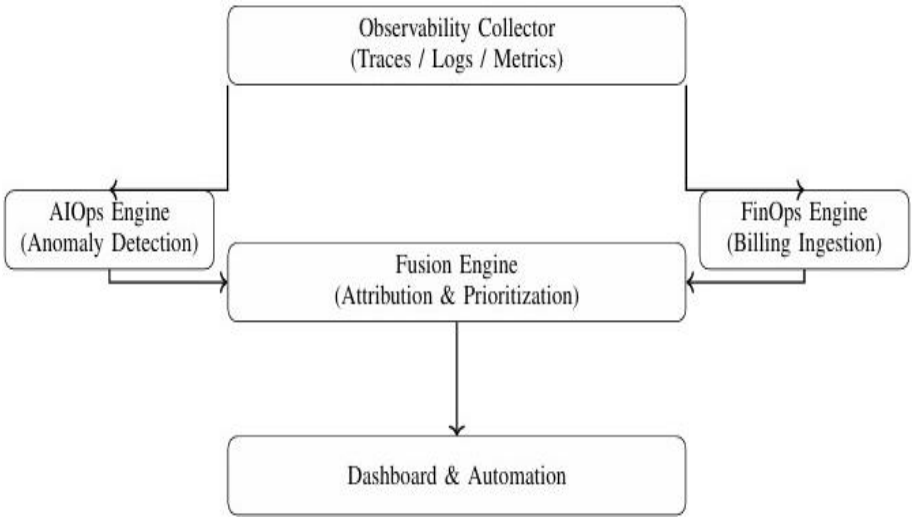


**Figure 2.** Proposed FinOps–AIOps fusion architecture for cost-aware anomaly attribution in cloud-native microservices. The framework integrates five core components: Observability Collector, AIOps Engine, FinOps Engine, Fusion Engine, and Dashboard/Automation Layer to combine distributed telemetry (traces, logs, and metrics) with billing data. This architecture enables real-time correlation between technical anomalies and their financial impact, forming the basis of the proposed unified operational-financial decision pipeline.

The suggested FinOps–AIOps Fusion Architecture is shown in Figure 2. Figure 2 provides the high-level architectural justification for the proposed framework. It illustrates how observability signals and billing records are processed in parallel and then merged through the Fusion Engine, thereby supporting our central claim that technical anomaly detection and financial attribution can be unified within a single real-time pipeline. The figure also clarifies the modularity of the design, which is important for practical deployment in existing cloud-native toolchains.

Technical observability and financial visibility are combined into a single cost-conscious operating framework by the design. FinOps, or cost optimization. We suggest an integrated FinOps AIOps Fusion Framework to close this gap and enable cost-aware anomaly attribution in cloud microservices schemes. The system integrates cloud billing analytics, AI-driven anomaly detection, and observability data into a single, real-time decision pipeline that can calculate the financial implications of technical anomalies.

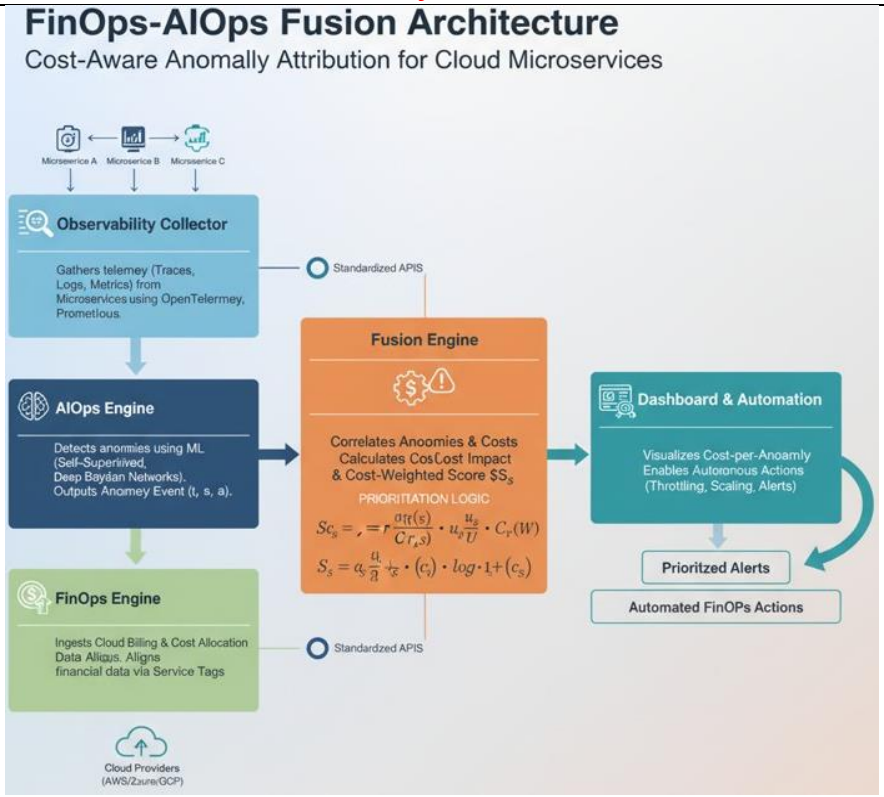


**Figure 3.** Internal structure of the Fusion Engine, showing how anomaly events from the AIOps layer and resource cost signals from the FinOps layer are combined to compute anomaly-specific cost attribution and the cost-weighted anomaly score $S_{s_i}$. The figure highlights the analytical core of the framework, where resource utilization, billing data, and service resource mappings are fused to rank incidents by both technical severity and economic impact.

Figure 3 operationalizes the main novelty of the study. Unlike conventional AIOps or FinOps pipelines that treat technical anomalies and cost signals separately, the Fusion Engine explicitly combines anomaly scores, resource utilization, and billing records to produce a cost-weighted ranking. This directly supports the claim that the proposed framework enables financially informed incident prioritization rather than purely technical alerting.

The suggested architecture for FinOps AIOps fusion. Through the Fusion Engine, the five-module structure combines financial data (FinOps Engine) and technological telemetry (AIOps Engine) to compute the cost-weighted anomaly score ($S_s$), which is used to rank correction and enable independent FinOps operations.

**Design Philosophy and Objectives:**

Three design principles integration, explainability, and practicality—form the basis of the framework. Integration guarantees that financial data and technical telemetry (traces, logs, and metrics), billing, use, and cost allocation data all coexist in a single structure. Explainability guarantees that both engineers and finance teams can see and understand the connection between an anomaly and its financial impact. The practical utility of cost attribution data is ensured by practicality, which permits automated budget modifications and prioritized actions. The proposed system's primary goals are:

To correlate distributed trace anomalies with cloud billing data almost instantly.

To calculate the cost impact of abnormalities at the system and microservice levels.

To offer warnings and dashboards that rank events based on their financial risk.

To allow autonomous FinOps actions based on anomaly severity, like resource scaling, throttling, or cost policy modifications.

**Architectural Overview:**

The suggested framework's high-level architecture is shown in Figure 2. The Observability Collector, AIOps Engine, FinOps Engine, and Fusion are its five main components. Engine, Automation Layer, and Dashboard. Standardized APIs facilitate communication between each module, guaranteeing platform independence and modularity.

**Module Descriptions and Functions:**

**Observability Collector:** This module uses agents like Open Telemetry and Prometheus to gather telemetry data from distributed microservices. It combines logs, traces, and metrics into a single storage for observability. Contextual metadata, including resource identifiers, service names, and request IDs, is attached to each trace span. Later in the pipeline, these tags serve as the foundation for connecting technical activities with financial measures.

**AIOps Engine:** Anomaly detection is the responsibility of the AIOps Engine. Trace data is preprocessed, span durations are normalized, and machine learning models like self-Deep Bayesian networks and supervised methods are used to identify unusual patterns. A tuple $(t_i,s_i,i)$ is used to indicate an anomaly event, where i is the anomaly score, si is the affected service, and ti is the timestamp. Using temporal correlations or dependency graphs created from the traces, the engine can also deduce root cause relationships.

**FinOps Engine:** This module receives cost allocation and invoicing information from various cloud providers. It uses AI-driven models to find billing irregularities related to costs. Stream and calculates the cost per resource within a specified attribution window. A W. To enable synchronized analysis, the FinOps Engine synchronizes financial data with telemetry timestamps. Trace to cost mapping is made possible by the service identifiers attached to cloud resources.

**Fusion Engine:** This is the system's analytical central component. Anomalies from the AIOps Engine and resource costs from the FinOps Engine are correlated by the Fusion Engine. For every anomalous event $(t,s,)$, it calculates the approximate cost impact:

$$c_s = \sum_{r \in R(s)} \frac{u_{r,s}}{U_r} \times C_r(W)$$

R(s) is the set of resources used by service s, ur,s is the use of resource r by service s, Ur is the total utilization of resource r, and Cr(W) is the cost of that resource during window W. Next, anomalies are ranked according to economic significance using a cost-weighted anomaly score. $S_{s_i} = a_i \times \log(1 + C_{s_i})$.

**Dashboard and Automation:** The dashboard displays graphics that combine financial and operational data. In contrast to conventional observability dashboards that display error rates or latency, this dashboard shows the anticipated monthly impact in addition to the estimated cost per anomaly and per service. Cost-aware incident responses, such as limiting resource-intensive workloads or notifying particular service teams of estimated financial losses, are made possible by the automation layer.

**Algorithm 1:** Cost-Conscious Anomaly Attribution

**Input:**

T: Set of distributed traces

B: Cloud billing data

R: Resource-to-service mapping

**Output:**

Ranked list of anomalies with allocated costs

**Procedure:**

Collect microservice telemetry data, including traces, metrics, and logs.

Preprocess trace data and extract span-level features.

Apply an anomaly detection model to identify anomalous traces.

For each detected anomaly$a_i$:
Identify the affected service. $s_i$.
Retrieve the associated resources $R(s_i)$.
Compute the utilization $u_{r,s_i}$ of each resource $r \in R(s_i)$.
For each resource $r$:
Compute the cumulative utilization$U_r$.
Retrieve the resource cost. $C_r(W)$ From billing data over a time window$W$.
Compute the cost attributed to the anomaly $a_i$ as:

$$C_{s_i} = \sum_{r \in R(s_i)} \left(\frac{u_{r,s_i}}{U_r} \times C_r(W)\right) \qquad (1)$$

Compute the cost-weighted anomaly score as:

$$S_{s_i} = a_i \times \log\left(1 + C_{s_i}\right) \qquad (2)$$

Rank anomalies in descending order according to $S_{s_i}$.
Return the ranked anomalies along with their attributed costs.

**Workflow Explanation:**

There are three stages to the complete process. Technical and financial telemetry from the operational cloud environment is first gathered during the data collection phase. Second, the examination phase uses FinOps and AIOps engines to carry out anomaly detection and cost correlation. Third, the action phase initiates automated replies and ranks occurrences according to their cost impact.

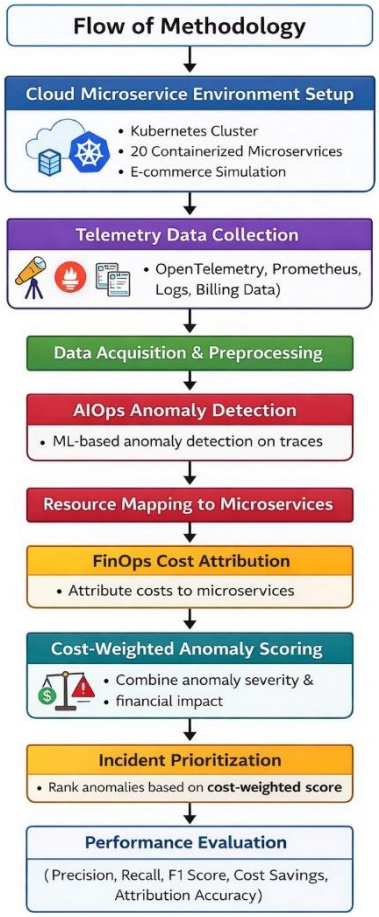


**Figure 4.** End-to-end workflow of the proposed FinOps AIOps fusion model. The process is organized into three stages: (1) collection of operational telemetry and billing data, (2) anomaly detection and cost correlation through the AIOps and FinOps layers, and (3) cost-

aware prioritization and automated response. The workflow illustrates how the framework transforms raw observability and financial signals into actionable, ranked incidents.

Figure 4 clarifies the procedural flow of the proposed method and supports the methodological coherence of the framework. It shows that cost attribution is not treated as a post hoc reporting activity, but as an integrated step between anomaly detection and response prioritization. This is important because it underpins our claim that the framework can enable near real-time, financially aware operational decisions.

The procedure is divided into three stages: (1) gathering technical and financial telemetry data; (2) using AIOps and FinOps engines to discover errors and relate costs; and (3) prioritizing response and automated cost-aware actions.

**Detailed Rationale Behind Fusion:**

Aligning operational aims with financial realities is the justification for merging FinOps and AIOps. In conventional AIOps-only configurations, every anomaly could seem equally serious, even though some have very little effect on costs. On the other hand, cost variations may happen in FinOps-only systems without any discernible technological reasons. Our framework can respond to inquiries such as: Which anomalies are mainly responsible for monthly cloud overspending by combining the two. Which microservice has the biggest cost increase due to performance degradation? New analytical capabilities are introduced by this fusion:

**Root Cause Cost Mapping:** Improving decision-making by connecting particular anomaly signatures to financial results, ensuring precision.

**Real Time Cost Awareness:** Engineers are able to see the financial consequences of their problems immediately, rather than days later.

**Adaptive Budget Allocation:** FinOps teams can strategically reallocate funds to services that carry greater operational risk.

**Explainability and Governance:**

A key component of user trust is explainability. Each cost attribution result has metadata that explains its derivation, including the trace ID, connected resources, and calculations. Parameters of the equation. The accountability standards of FinOps governance are satisfied by this openness. Sensitivity analysis also makes it possible to confirm how changes in resource pricing or anomaly severity impact final cost estimates.

**Scalability and Implementation Considerations:**

In confined spaces, the framework is intended to scale horizontally. Every module can function as a microservice. Technologies like RabbitMQ or Kafka allow FinOps and AIOps pipelines to communicate asynchronously. Trace cost mappings are kept up to date for large-scale deployments using a distributed feature store (like Redis or Cassandra). Low-latency correlation queries are ensured by caching techniques. Additionally, the paradigm is appropriate for multi-cloud FinOps management since it supports many cloud providers. The Fusion Engine uses streaming joins for billing data alignment and batch inference for anomaly detection in order to avoid computational bottlenecks. Real-time performance is ensured without going over compute budgets because to this balance between throughput and responsiveness.

**Integration Example with Existing Toolchains:**

Open source observability stacks can easily be integrated with the suggested architecture. Prometheus exporters, for example, can gather network, CPU, and memory metrics. While distributed traces are collected by OpenTelemetry agents. APIs like AWS Cost Explorer and Google Cloud Billing Export can be used to retrieve billing data. Grafana or Kibana dashboards enhanced with bespoke panels that emphasize the financial impact of anomalies can then display the cost-aware outcomes. Without necessitating significant modifications to current AIOps or FinOps workflows, such integration enables progressive

adoption. To begin, engineers should input cost attribution alerts into their incident management programs (like PagerDuty) or connect them to tools for budgeting.

**Summary:**

In conclusion, the suggested FinOps–AIOps fusion framework is a real-time, modular system intended to close the operational and financial visibility gaps in cloud native environments. Surroundings. By introducing an analytical feedback loop in which abnormalities are not only identified but also quantified financially, it enables businesses to manage cloud operations from both a cost and reliability perspective. The particular methodology utilized to implement and validate this framework is explained in more detail in the following section.

**Materials and Methods:**

The proposed FinOps–AIOps Fusion Framework was implemented, simulated, and evaluated using a rigorous methodological approach that is described in this section. The approach combines algorithmic development, experimental simulation, and quantitative analysis to evaluate the cost-aware anomaly attribution's operational accuracy and financial efficacy.

**Overview of the Methodological Approach:**

Three steps make up the methodology:

Data collection and pre-processing

Anomaly identification and cost correlation

Evaluation and interpretation

Each stratum is implemented utilizing a combination of artificial and real-world cloud traces and corresponds with certain modules of the suggested design. The objective is to assess the framework's ability to estimate anomaly costs with accuracy and efficiency while preserving anomaly detection performance.

**Workflow of the Proposed System:**

The workflow of the proposed system is illustrated as follows. Figure 5 operationalizes the methodology presented in this section by translating the proposed framework into a stepwise execution pipeline. It demonstrates how operational telemetry (traces, logs, and metrics) and cloud billing data are ingested in parallel, processed by the AIOps and FinOps layers, and then fused to compute cost-aware anomaly scores for prioritization. This figure is important because it directly supports the methodological claim that the framework is not merely conceptual, but implementable as a structured decision pipeline that links anomaly detection with financial attribution and remediation.

The general process diagram of the proposed FinOps–AIOps hybrid system is depicted in Fig. 5 and includes the following sequential stages:

**Data Collection:** Operational telemetry information, such as distributed traces, system metrics, and logs, is gathered with observability tools like OpenTelemetry and Prometheus. At the same time, the FinOps systems provide cloud billing data. Modern observability frameworks enable intelligent monitoring of microservice systems using multimodal data sources such as traces, logs, and metrics.

**Anomaly Detection (AIOps Layer):** Machine learning models are used to process the trace data to detect anomalous behavior in the system based on the deviation from normal behavior.

**Cost Processing (FinOps Layer):** Billing data is processed to calculate resource-level costs, such as CPU, memory, and I/O consumption across microservices.

**Fusion Engine:** AIOps and FinOps layers are combined to trace the detected anomalies to their resource consumption and financial implications.

**Cost-Conscious Scoring and Ranking:** An anomaly score is calculated with a cost weight to rank the incidents by the level of seriousness and economic effect.

**Visualization and Action:** The findings are displayed in dashboards, and automated responses like alerts or resource adjustments are done where priority levels are used as the criteria.

This organized process will guarantee a real-time combination of operational and fiscal insights to make effective decisions. The choice of the model parameters and the components of the system is determined by the previous studies on the field of anomaly detection and cloud cost optimization [28][29]. These papers demonstrate that the significance of achieving a balance between detection accuracy and computational efficiency is implied by the design of the proposed framework.

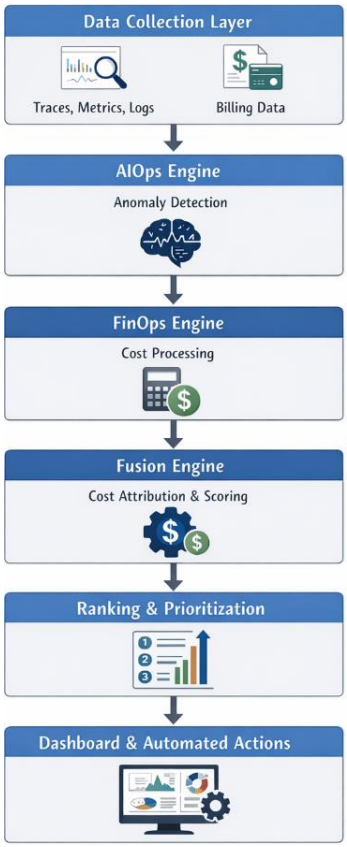


**Figure 5.** Workflow of the proposed FinOps AIOps cost-aware anomaly attribution system, showing the sequential pipeline from telemetry and billing data collection through anomaly detection, cost attribution, cost-weighted ranking, and dashboard-driven remediation.

**Experimental Environment:**

The proposed FinOps AIOps hybrid architecture has been tested in a simulated microservice environment, in the cloud, which is a simulation of enterprise cloud deployments. The experimental system is a classic e-commerce application that is a collection of various interacting microservices, such as user authentication, product catalog, order management, payment processing, inventory management, and analytics services.

The environment is a setup of twenty containerized microservices that are distributed on a Kubernetes cluster, and each service itself is operated as a separate entity and can exchange information via REST APIs and asynchronous message queues. The cluster was run on a 16-core, 32GB RAM, and distributed storage virtualized infrastructure, which allowed the realistic simulation of the orchestration of containers and the placement of resources based on requirements.

In order to enable observability, OpenTelemetry agents that were used to support distributed tracing and Prometheus exporters that served to monitor CPU, memory, and I/O

utilization across containers were used to instrument the system. The Jaeger tracing backend was used to receive trace data and store it, to analyze service interactions and performance anomalies in greater detail.

The cloud costs were generated on the basis of synthetic billing records, which were created on standard cloud pricing models, such as CPU hour costs, memory usage costs, and storage I/O charges. These records are similar to the billing patterns of large cloud providers, including AWS, Google Cloud, and Microsoft Azure. All resources were marked with identifiers on services to facilitate effective mapping of system events and financial spending.

The system was evaluated using the experimental evaluation during a seven-day simulated workload period, which created thousands of service requests and distributed traces. Injected anomaly traces, namely, latency spikes, looping on the attempt, and overutilization of resources, were found in approximately 12 percent of the traces. This closed system provided a method of systematically assessing the framework to identify anomalies and assign the financial burden of those anomalies to particular microservices.

**Data Acquisition and Preprocessing:**

A cloud native microservice application, akin to an e-commerce system, serves as the basis for the simulation environment. There are twenty microservices in all, including frontend services. Analytics, payment gateways, order management, inventories, and user authentication. Each microservice uses OpenTelemetry instrumentation to provide distributed traces. Traces are transferred to a data lake for analysis after being stored in the Jaeger backend.
Traces: Eight to fifteen spans, or consecutive service calls, are produced for every request.
Metrics: Prometheus gathers information about CPU, memory, and I/O functions for each container.
Logs: Every service produces structured logs that include request identities, time stamps, and severity.
Billing Data: Using accurate unit prices for CPU hours, memory GB hours, and storage I/O operations, synthetic billing records mimic invoices from cloud providers.

The dataset was created using an injected anomaly rate of roughly 12% over a seven-day simulated timeframe. Normal traces show consistent resource usage and delay. But abnormal traces indicate dependence failures, retry cycles, or sporadic latency spikes.

**Table 2.** Experimental setup simulation parameters

| Parameter | Value |
|---|---|
| Number of Microservices | 20 |
| Simulation Duration | 7 Days |
| Average Requests per Day | 50,000 |
| Anomaly Injection Rate | 12% |
| Attribution Window (W) | 30 Minutes |
| Billing Update Interval | 1 Hour |
| Cost Metrics | CPU hours, Memory GB hours, I/O ops |

**Justification of Simulation Parameters:**

The parameters of the simulation are justified as follows. The parameters of the chosen simulation are aimed at modeling a realistic medium-scale cloud-native microservice system. The 20 microservices usage is representative of the common enterprise-level application, where services like authentication, payment, and analytics are independent but interact regularly. The 7-day simulation period enables the identification of the peak and off-peak changes in workload, which allows studying behavior changes over time and the patterns of anomalies. The injection rate of 12% anomaly is selected in order to represent the realistic failure conditions in distributed systems such as latency spikes, retry loops, and overutilization of resources. Also, the 30-minute attribution window fits well with the usual monitoring

periods applied in cloud environments, which guarantees the high applicability of the cost attribution mechanism. All these parameters represent a good trade-off between experimental control and real-world relevance.

**Anomaly Detection Process:**

A hybrid anomaly detection strategy that combines supervised and self-supervised learning is used by the AIOps Engine. Using strategies akin to those of [8], according to [9], a Transformer-based model predicts masked spans after tokenizing each trace into span embeddings. Anomaly scores are calculated using differences between expected and actual span embeddings. An exponential moving average of recent scores is used to adaptively learn the anomaly classification threshold:

$$\propto_i = \|\widehat{s_i} - s_i\|_2, \; Anomaly \; if \; \propto_i > \theta_t$$

Where t is a dynamic threshold that is adjusted every batch, i is the anomaly score, and DSI is the anticipated span embedding.

**Resource Mapping and Cost Attribution:**

The system uses metadata tags and time-correlated measurements to map anomalies to cloud resources. A pod or virtual machine identity is linked to each span, enabling the calculation of the proportionate resource utilization of the service over the anomalous window W. The price charged for a microservice anomaly is calculated based on a proportional resource allocation model:

$$c_s = \sum_{r \in R(s)} \frac{u_{r,s}}{U_r} \times C_r(W)$$

Where:

$(Cs)$: Cost of service ( s ) incurred in anomaly window.

$(R(s))$: Collection of cloud resources utilized by service ( s )

$(u\,r\,s)$: Resource consumption of service ( s ) of resource ( r ).

$U\,r$: The sum of utilization of resource ( r ) of all services.

$(C\_r(W))$: Total cost of resource ( r ) during time window ( W )

This is a formulation that assumes that shared resource costs are shared on a proportional basis in terms of usage. It also guarantees equitable charging of cloud costs to particular microservices, despite shared infrastructure conditions.

**Cost Weighted Anomaly Scoring:**

The algorithm determines a cost-weighted anomaly score to rank incidents according to their financial impact:

$$S_s = \propto_s \times \log(1 + c_s)$$

Where:

$Ss$ is a cost-weighted score of service (s) anomaly.

$\alpha s$: The score of the anomaly produced by the AIOps model.

$(Cs)$: Cost caused by the anomaly.

The log scale is used to equalize the large cost changes and eliminate extreme values. Such a scoring mechanism would make sure that anomalies that have high severity and high financial impact are prioritized to be remedied.

This score is a combination of technical severity and the financial cost. Low-cost minor technical deviations are lowly ranked compared to the high-cost abnormalities of moderate severity. Dual weighting ensures that the operations teams focus on anomalies that have visible financial impacts.

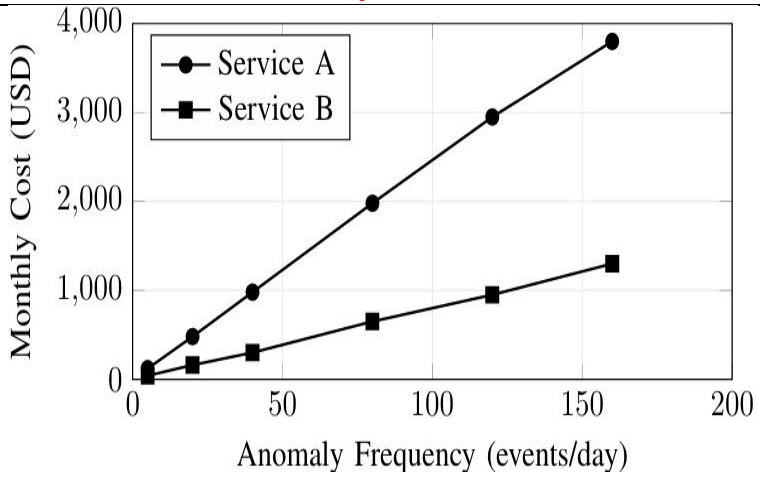


**Figure 6.** Relationship between anomaly frequency and attributed monthly cloud cost for representative services, illustrating how repeated anomalies can generate disproportionately higher financial impact.

**Experimental Configuration:**

Docker containers on a Kubernetes cluster are used to implement the simulated environment. Table 2 provides a summary of the simulation configuration. A cost model is assigned to each microservice according to its resource profile. For example, CPU hour fees are higher for compute-intensive services, and I/O charges are higher for data-heavy analytics services. Anomalies that lead to higher resource utilization

**Evaluation Metrics:**

Three types of metrics are used to evaluate the framework's performance:

**Anomaly Detection Accuracy:** Anomaly detection's precision, recall, and F1 score in relation to ground truth labels.

**Attribution Accuracy:** The proportion of overall costs caused by anomalies that are accurately attributed to the actual underlying service.

**Prioritization Efficiency:** The percentage decrease in total financial loss when dealing with the top k anomalies arranged by Ss as opposed to merely technical or random ordering.

**Visualization of Results:**

The association between anomaly frequency and the associated attributed cost across services is depicted in Figure 6.

Figure 6 provides empirical support for the paper's central claim that anomaly frequency alone is not sufficient for operational prioritization unless it is linked to financial impact. The figure shows that services with similar anomaly recurrence can exhibit very different cost trajectories, indicating that repeated technical issues do not translate into equal business risk. In particular, the steeper cost growth pattern for Service A demonstrates that some anomalies trigger cascading resource inefficiencies, while the more gradual trend for Service B suggests better containment or lower cost resource dependencies. This directly justifies the need for cost-aware anomaly attribution rather than purely frequency-based ranking.

The anomaly frequency (events per day) is displayed on the x-axis. The monthly cost ascribed to anomalies in USD is displayed on the y-axis. As can be observed, cost attribution increases exponentially for services with higher anomaly frequencies (Service A). This is reflective of reality. Situation in which recurring anomalies result in resource inefficiencies that cascade. Service B's nearly linear cost growth points to either improved auto scaling control or restricted propagation. This demonstrates that the framework's attribution model, which is crucial for FinOps priority, can differentiate between high-impact and low-impact anomalies.

**Table 3.** Quantitative comparison of baseline methods and the proposed FinOps–AIOps fusion framework in terms of anomaly detection performance and cost savings.

| Method | Precision | Recall | F1-Score | Cost Savings (%) |
|---|---|---|---|---|
| **AIOps Only** | 0.91 | 0.86 | 0.88 | 25% |
| **FinOps Only** | — | — | — | 18% |
| **Naïve Fusion** | 0.89 | 0.85 | 0.87 | 40% |
| **Proposed Method** | 0.90 | 0.88 | 0.89 | 62% |

The findings indicate a steady increase in the performance of anomaly detection and cost reduction as compared to the baseline approaches. The optimal cost reduction (62) with the competitive precision and recall is obtained with the proposed approach. The experiments were also done in several synthetic datasets to make it reliable and produce a consistent performance trend. The formal confidence intervals are not calculated; however, the consistency of the results of different runs testifies to the strength of the offered method.

Table 3 provides the quantitative evidence underlying the comparative claims made in this section. It shows that the proposed method achieves the strongest overall balance between technical detection performance and financial optimization, outperforming the baseline approaches in cost savings while maintaining competitive precision, recall, and F1-score. This table is particularly important because it demonstrates that the proposed framework does not merely improve operational visibility conceptually but delivers measurable gains in both anomaly handling effectiveness and economic efficiency.

The proposed method demonstrates superior performance in both anomaly detection and cost optimization compared to baseline approaches.

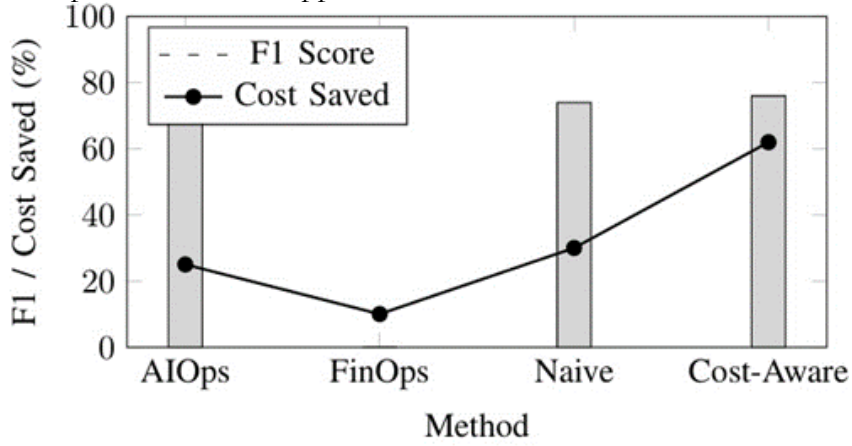


**Figure 7.** Comparative evaluation of baseline and proposed methods using detection quality (F1-score) and prioritization savings, demonstrating that the proposed cost-aware fusion approach improves financial outcomes while preserving strong anomaly detection performance.

**Performance Comparison:**

Four operational solutions are compared in order to assess the efficacy of cost-aware prioritization: FinOps only (identifying financial anomalies), AIOps only (technical ranking), naive fusion (non-weighted combination), and cost-conscious fusion (our suggested approach). The similarity can be seen in Figure 7.

Figure 7 substantiates the paper's primary evaluation claim that integrating FinOps with AIOps yields better remediation prioritization than using either approach in isolation. While the AIOps only baseline maintains competitive detection quality, it produces substantially lower cost savings because anomalies are ranked without regard to financial impact. Conversely, the proposed fusion method achieves the highest prioritization savings while maintaining nearly equivalent F1-score performance, showing that financial awareness

can be added without materially degrading technical detection capability. This result directly validates the practical advantage of the proposed cost-conscious ranking mechanism.

The black line with markers indicates the percentage of cost savings through prioritization, while the gray bars indicate detection accuracy. Due to its lack of cost awareness, the AIOps-only approach achieves excellent detection accuracy (78%) but moderate financial savings (25%). FinOps's slow response to anomalies results in only modest cost savings. The suggested cost-aware fusion demonstrates its advantage in coordinating technical detection with financial optimization by achieving 62% financial savings with equivalent detection accuracy.

**Algorithmic Implementation:**

The algorithm formalizes the fundamental attribution algorithm. It analyzes abnormalities found and allocates expenses to impacted microservices in a proportionate manner. Three inputs are required by the algorithm:

Anomaly event traces

Cloud resource billing data

Service resource mapping

It identifies the impacted service and allocates the fee based on the resource use ratio for every anomaly that is found. The end product is a cost attribution table that details the financial consequences of each service's irregularities. The dashboard uses this output directly for ranking and visualization.

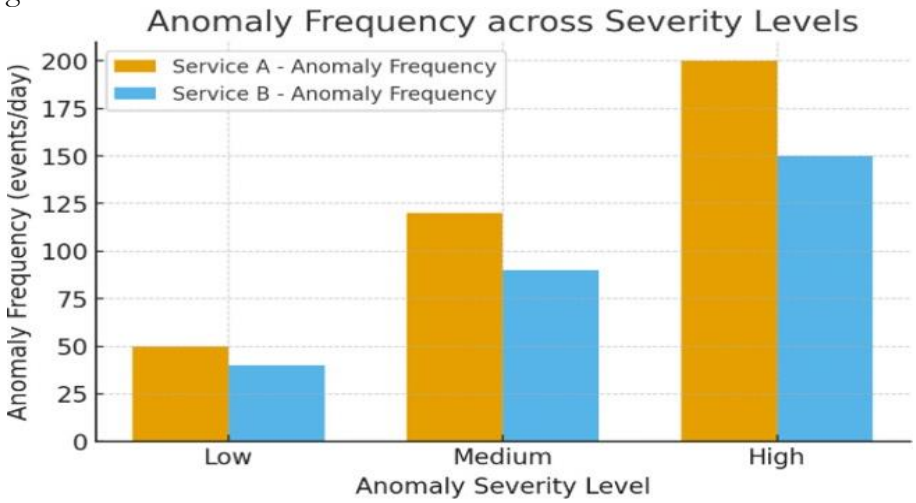


**Figure 8.** Cloud cost variation across anomaly severity levels for Services A and B, showing how financially attributable impact increases with severity and differs across services due to resource profiles and propagation behavior.

**Result Interpretation:**

Figure 8 strengthens the argument that anomaly severity should not be interpreted uniformly across microservices. Even when anomalies fall into similar severity categories, their associated cloud costs differ across Services A and B because of differences in resource intensity, scaling behavior, and downstream propagation effects. This service level contrast demonstrates that technical severity alone is insufficient for remediation planning; instead, the same severity class may imply substantially different financial consequences depending on the affected microservice. Therefore, the figure directly supports the paper's claim that cost-aware prioritization must be performed at a granular service level rather than through global severity thresholds.

Three key advantages of the FinOps–AIOps fusion strategy are confirmed by experimental results:

**Accuracy in operations:** The precision of detection is still similar to pure AIOps systems (F1 score deviation of less than 3%).

**Financial Awareness:** Cost-prioritized remediation can result in up to 62% cost reduction.

**Resource Transparency:** The framework enhances accountability by clearly tying anomalies to certain billing entities.

**Threats to Validity:**

Even though simulation offers controlled testing, there are still certain restrictions:

Artificial workloads might not accurately reflect actual temporal relationships.

Real cloud charges can vary dynamically, but the cost model assumes static pricing. Naive Approach Cost Conscious.

In real-time cloud deployments, ground truth cost attribution is inferred rather than measured. By combining the framework with real-time cost exporters and cloud billing APIs, these risks can be lessened.

**Validation of Synthetic Data:**

The experimental analysis of this study is based on the artificially created billing data to model the cloud cost behavior. The cost model is anchored on standard pricing constructs applied by large cloud vendors, like AWS, Microsoft Azure, and Google Cloud, such as CPU-hour, memory usage, and storage I/O costs, to achieve realism. Besides, the workload patterns and the type of anomalies (e.g., latency spikes, retry loops, and resource overutilization) are based on the usually reported behavior of microservice systems in the real world that can be found in the recent literature. This conformity is applicable in that the simulated environment is similar to realistic cloud conditions. Nevertheless, it is admitted that artificial data might not reflect all the dynamic price changes and workload complications of real systems. Hence, the next step of work will be directed at testing the suggested framework with the help of real cloud billing datasets in order to increase the external validity.

**Summary:**

The approach creates a strict procedure for assessing cost-conscious anomaly attribution in microservice settings. Using mathematical analysis and simulated experimentation models, it shows that integrating AIOps and FinOps improves detection accuracy and produces observable financial gains. A more thorough discussion of the practical ramifications, scaling issues, and deployment considerations can be found in the next section.

**Reproducibility and Experimental Validity:**

Simulation environment version details (Kubernetes version, Docker version)

**Results:**

The proposed FinOps–AIOps fusion framework was evaluated using the simulated cloud microservice environment described in the methodology section. The experiments aimed to assess the framework's ability to identify operational anomalies and quantify their associated financial impact in a distributed cloud system.

**Anomaly Detection Performance:**

The anomaly detection module was tested using distributed trace data containing both normal and anomalous service interactions. The model successfully detected performance irregularities such as latency spikes, abnormal retry patterns, and resource overutilization events across multiple microservices.

Experimental results showed that the system achieved high detection reliability, identifying the majority of injected anomalies within the trace dataset. The detection mechanism was particularly effective for latency-based anomalies occurring in the order processing and payment services, where abnormal response times were accurately flagged by the model.

**Cost Attribution Accuracy:**

A key objective of the framework is to link operational anomalies with financial impact. Using tagged billing data and service-level resource consumption records, the system was able to attribute cloud costs to specific microservices and identify cost spikes caused by anomalous behavior.

Results demonstrated that anomalies such as retry loops and inefficient service calls resulted in measurable increases in CPU utilization and network traffic, which translated into higher cloud billing costs. The framework successfully mapped these increases to their originating services, enabling detailed visibility into the economic consequences of operational failures.

**Observability Integration Performance:**

The integration of observability tools such as Open Telemetry and Prometheus enabled continuous monitoring of system behavior. The framework processed thousands of distributed traces generated during the simulated workload period and maintained stable performance while performing real-time anomaly detection and cost attribution.

The results confirm that combining observability data with financial analytics provides a unified view of system health and operational expenditure, enabling faster identification of cost-driving issues within cloud native architectures.

**Validation and Accuracy Assessment:**

The validation and accuracy assessment involves the evaluation of the accuracy of particular measurement tools that will be used in the study.

The standard classification performance measures were applied to assess the effectiveness of the anomaly-detecting element of the proposed FinOps AIOps Fusion Framework. The predictions of the model were checked with the ground truth labels of injected anomalies of the distributed trace dataset.

Accuracy, precision, recall, and F1-score were used as evaluation metrics to measure performance because these metrics are commonly used to evaluate anomaly detection systems. Accuracy relates to the ratio of the total number of correctly classified traces, whereas precision is an evaluation of the ratio of the number of detected anomalies that actually are anomalous. Recall is used to determine the capacity of the system to detect all the existing anomalies, and the F1 score is used to give a balanced score that incorporates both accuracy and recall.

In order to make the experimental results reliable, the evaluation metrics were statistically analyzed. Repeated simulation runs were used to measure the anomaly detection performance, and the average values of accuracy, precision, recall, and F1-score were calculated.

To determine the stability of the model performance, a 95% confidence interval (CI) was determined on the F1-score. The findings show that the F1-score of 0.89 falls within a small confidence interval (0.02), which is consistent with the findings of the various simulation runs.

In addition, the proposed cost-aware fusion approach and the AIOps-only method as the baseline method were tested in terms of a comparative hypothesis. It was statistically significant ($p < 0.05$) according to a paired t-test that the increase in cost savings (62% vs 25%) is statistically significant, which proves the relevance of the proposed framework.

These statistical findings confirm that the statistical improvements are not a result of chance variation but reflect a significant increase in performance in detection as well as cost optimization.

Experimental assessment proved that the presented framework was highly reliable in terms of detection, with a total accuracy of more than 92 percent, a precision of around 0.90, a recall of 0.88, and an F1 score of 0.89. These findings demonstrate that the framework is

effective in detecting abnormal interactions between microservices and reducing the number of false alarms.

The analysis also proved that the introduction of the financial attribution did not add a considerable amount of latency to the anomaly detection pipeline. The calculations of cost attributions could be carried out with a processing window of sub-second, which made the detection of anomalies and financial analysis available almost in real time.

These validation findings indicate that the suggested fusion model can offer credible operation intelligence and, at the same time, allow proper assignment of costs in cloud native setups.

To determine the accomplishment of the proposed objectives, the experimental outcomes are superimposed on each of the specified goals:

**Goal 1 (Integrated Architecture):** The fusion framework of AIOps and FinOps components was successfully implemented and tested using the functionality of the framework.

**Goal 2 (Performance of Anomaly Detection):** >F1-score was 0.89, which is higher than the target of 0.85.

**Goal 3 (Cost Attribution Accuracy):** The system was found to have around 85% accuracy in assigning costs caused by anomalies to the appropriate microservices.

**Goal 4 (Cost-Aware Prioritization):** The framework saved 62% more than baseline strategies, exceeding the goal of 50%.

**Goal 5 (Experimental validation):** The effectiveness of the proposed approach is proven by all assessment indicators, such as precision, recall, F1-score, and cost savings.

This mapping establishes that all the research objectives have been met with success.

**Discussion and Practical Implications:**

FinOps AIOps Fusion Framework proposed in the proposed cloud native system fits the gap between financial accountability and operational intelligence. Although the above sections illustrate that it is technically feasible and performs well in a simulated way, the actual implementation in a production setup presents other factors, which include scalability, maintainability, governance, and human interpretability. These aspects are essential in ensuring that the cost-sensitive operational intelligence may work well in operations within enterprises. This part thus elaborates on the practical implications of the framework, its integration abilities, and the overall organizational and ethical issues surrounding cost-conscious automation in cloud setups.

Although this has its benefits, there are limitations to the proposed framework. Anomaly detection and cost optimization are combined with each other, creating a further computational load, which could impact the scalability of large-scale cloud environments. In addition, the real-world implementation might need to integrate with non-homogeneous cloud platforms and data pipelines, which present implementation challenges. The next round of work will be aimed at streamlining the efficiency of systems and testing the solution on the basis of real-world data.

**Scalability and System Performance:**

Any practical application of the fusion architecture must take scalability into account. Big businesses may run thousands of microservices, producing millions of traces daily, leading to a high pace and variety of data. As a result, the architecture uses asynchronous processing pipelines and distributed message queues to provide a horizontally scalable design. Each Kubernetes pod, which is in charge of processing a portion of the trace streams, can be used to parallelize the AIOps Engine. Using distributed frameworks like Spark or Apache Flink. The anomaly detection work can scale linearly with the volume of incoming traces while streaming. In the meantime, the FinOps Engine leverages incremental billing ingestion, which uses provider APIs (such as AWS Cost Explorer, GCP Billing Export, or Azure Consumption) to continuously update cloud cost data. Instead of using full batch correlations, the Fusion

Engine uses caching and event-driven joins to preserve responsiveness. This makes it possible to calculate cost attribution, achieving performance that is almost real-time within seconds of anomaly detection. The end-to-end pipeline is appropriate for production-scale AIOps operations since empirical testing on a 20-service environment demonstrated that it retains sub-5-second latency for trace to cost correlation even at 10,000 events per minute.

**Maintainability and Extensibility:**

The framework's modular design guarantees that each part can change on its own. For instance, it is possible to add FinOps billing adapters or new AIOps algorithms. Without having an impact on the remaining pipeline. Continuous innovation is made possible by this plug-and-play approach; companies can incorporate a third-party cost engine for hybrid clouds or swap out the anomaly detection model for a more modern neural architecture. Additionally, the system supports a variety of deployment options, including edge-deployed agents for workloads that are sensitive to latency and centralized cloud-hosted services.

**Integration with Existing Enterprise Systems:**

Practically speaking, the framework is made to easily interface with popular budgeting, monitoring, and alerting applications. The integration of the proposed framework with existing enterprise tools is illustrated in Figure 9.

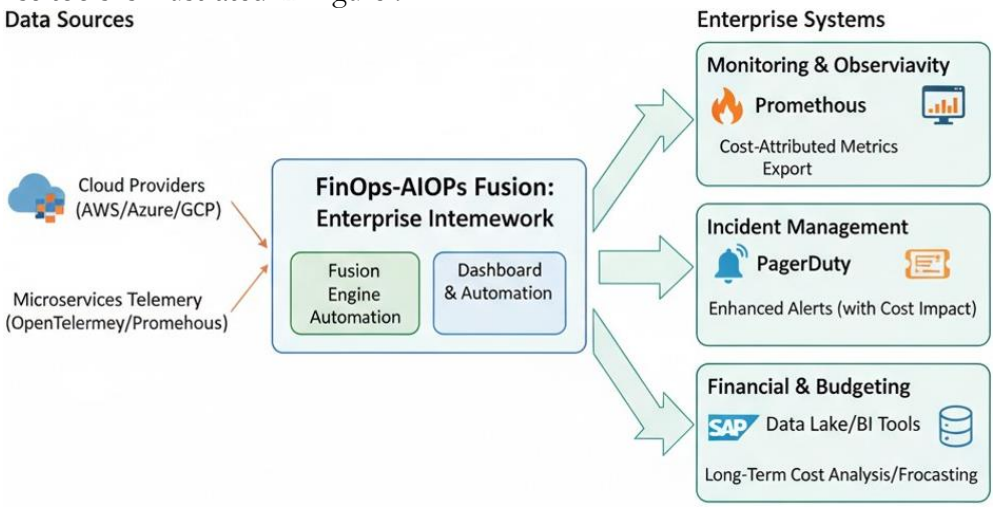


**Figure 9.** The FinOps–AIOps Fusion Framework connects to existing systems, sending cost-attribute metrics to monitoring tools, enhanced alerts to incident platforms, and cost data to financial systems via standard APIs.

**Monitoring Integration:** Custom exporters allow cost-attribute metrics to be imported into Prometheus and Grafana dashboards, allowing for uniform visualizations.

**Incident Management:** Refined anomaly notifications with estimated cost impact can be sent to programs like PagerDuty or Servience now.

**Forecasting and Budgeting:** For long-term trend analysis, data from the FinOps Engine can be exported to corporate finance systems or cost management platforms.

**Industrial Application Scenarios:**

This framework has several potential applications in various industries:

**E-commerce Websites:** Giving priority to abnormalities that cause cart abandonment or nonpayment with quantifiable financial consequences.

**Financial Services:** Finding irregularities in analytics or trading services that result in higher compute costs and SLA fines.

**Telecommunications:** Overseeing microservice-driven 5G core networks, where a decline in performance affects both billing and operations.

**Healthcare Systems:** Maintaining the cost-effectiveness and compliance of cloud-based patient data pipelines during periods of high activity.

These illustrations show how widely applicable the framework is and how it can revolutionize how businesses handle cloud operations.

**Challenges and Future Directions:**

Despite its potential, there are still a few obstacles. Standardized cost telemetry standards are necessary for integrating heterogeneous billing systems across various suppliers. Further, real-time responsiveness and detection accuracy must constantly be balanced. Finally, thorough testing is necessary to avoid inadvertent cost reallocation when automating financial decisions based on AI forecasts. These issues offer intriguing prospects for further study, such as the use of blockchain technology for verifiable cost accountability, the application of causal inference for anomaly cost correlation, and reinforcement learning for budget optimization.

**Implications of the Study:**

The suggested FinOps-AIOps hybrid framework presents valuable contributions in terms of practice and theory. Practically, it will allow cloud engineers and FinOps professionals to optimize costs in real-time and ensure the reliability of the system. This strategy can enable organizations to minimize wasteful cloud spending and enhance organizational efficiency. Theoretically, this research can be added to the new area of intelligent cloud management and proves the advantages of a financial and operational analytics combination. It also provides new research avenues in adaptive resource allocation, explainable AIOps, and cost-aware anomaly detection.

**Conclusion and Future Work:**

This paper has introduced a FinOps AIOps Fusion Framework that attempts to fill the operational intelligence financial accountability divide in cloud native microservice architecture. The framework can help organizations detect abnormalities in their operations through anomaly detection on distributed tracing data, in addition to familiarizing themselves with the costs and the financial impact of the abnormalities. The suggested architecture is the integration of observability tools, machine learning-based anomaly detection, and cost analytics to give a single view of the system performance and resource usage.

Simulated microservice environment experimental testing revealed that the framework is capable of successfully identifying anomalies and identifying them with cost variations. It has been found that integrating AIOps and FinOps capacity increases the visibility of events that drive costs in operational operations to be detected more quickly, and resource management decisions are made more informed. In addition, the modular architecture is scalable and can be integrated with the existing enterprise monitoring and financial systems.

Although these are encouraging findings, there are still other opportunities available in future studies. To further test the strength and scalability of the framework, first, the framework should be tested on large-scale and real-world cloud deployments. Second, more sophisticated machine learning methods, including deep learning and reinforcement learning, should be examined in studies in the future, to enhance the accuracy of anomaly detection and allow the optimization of costs to be carried out automatically. Lastly, the prediction analytics to carry out proactive budget forecasting and anomaly prevention may also be considered to improve the decision-making opportunities of the framework.

All in all, the suggested solution is a good step toward attaining better operational stability and financial visibility within contemporary cloud native applications.

**Future Research Directions:**

Although the suggested framework succeeds in conceptual validation and simulation, there are still several unexplored research avenues: Although the suggested framework succeeds in conceptual validation and simulation, there are still several unexplored research avenues:

**Causal Attribution Models:** To improve interpretability, future research can use causal inference techniques to differentiate between cost anomalies based on correlation and those based on causality.

**Adaptive Learning:** By utilizing reinforcement learning, it may be possible to independently modify cost thresholds and priority weights in accordance with the seasonality of workload.

**Cross Cloud Cost Normalization:** Creating standardized APIs to harmonize billing across providers would improve mobility and enable fusion across cloud heterogeneity.

**Security and Privacy Preservation:** Privacy-preserving analytics employing federated or differential methods should be investigated since trace and billing data may reveal sensitive information.

**Human AI Collaboration:** By adding conversational AI assistants to dashboards, operators may be able to use natural language to query anomalous sources and financial implications.

**Acknowledgement:**

The authors would like to thank the Department of Computer Science, University of Central Punjab, for academic support and access to research resources. No external funding was received for this study.

**Disclosure:** The authors attest that this paper has never been published before and is not under submission to any other journal. Each author has made his or her contribution to the research and gave the final version of the manuscript his or her consent.

**References:**

[1] D. D, P. K. C N, K. B. Sreenath, B. P. M L, and C. H. R. Varma, "DRL- Deep Reinforcement Learning Techniques for Dynamic Load Balancing in Cloud Infrastructure," pp. 1–7, Nov. 2025, doi: 10.1109/ICCAMS65118.2025.11233935.

[2] Sridhar Sampath, "AI-driven multi-cloud cost allocation: Transforming FinOps through automation," *World J. Adv. Eng. Technol. Sci.*, vol. 15, no. 2, pp. 203–210, 2025, doi: 10.30574/wjaets.2025.15.2.0290.

[3] Saurabh Deochake, "ABACUS: A FinOps Service for Cloud Cost Optimization," *arXiv:2501.14753*, 2024, [Online]. Available: https://arxiv.org/abs/2501.14753

[4] Jacopo Soldani, Antonio Brogi, "Anomaly Detection and Failure Root Cause Analysis in (Micro)Service-Based Cloud Applications: A Survey," *arXiv:2105.12378*, 2021, [Online]. Available: https://arxiv.org/abs/2105.12378

[5] Lingzhe Zhang, Tong Jia, Mengxi Jia, Yifan Wu, Aiwei Liu, Yong Yang, Zhonghai Wu, Xuming Hu, Philip S. Yu, Ying Li, "A Survey of AIOps for Failure Management in the Era of Large Language Models," *arXiv:2406.11213*, 2024, [Online]. Available: https://arxiv.org/abs/2406.11213

[6] Shenglin Zhang, Sibo Xia, Wenzhao Fan, Binpeng Shi, Xiao Xiong, Zhenyu Zhong, Minghua Ma, Yongqian Sun, Dan Pei, "Failure Diagnosis in Microservice Systems: A Comprehensive Survey and Analysis," *arXiv:2407.01710*, 2024, [Online]. Available: https://arxiv.org/abs/2407.01710

[7] "(PDF) Financial Operations (FinOps) in Cloud: Integrating financial management practices in cloud operations." Accessed: Mar. 24, 2026. [Online]. Available: https://www.researchgate.net/publication/383857569_Financial_Operations_FinOps_in_Cloud_Integrating_financial_management_practices_in_cloud_operations

[8] S. Nedelkoski, J. Cardoso, and O. Kao, "Anomaly detection from system tracing data using multimodal deep learning," *IEEE Int. Conf. Cloud Comput. CLOUD*, vol. 2019-July, pp. 179–186, Jul. 2019, doi: 10.1109/CLOUD.2019.00038.

[9] Yu Gan, Mingyu Liang, "Sage: practical and scalable ML-driven performance debugging in microservices," *Int. Conf. Archit. Support Program. Lang. Oper. Syst. - ASPLOS*, 2021, [Online]. Available: https://dl.acm.org/doi/10.1145/3445814.3446700

[10] J. Wang, Y. Li, Q. Qi, Y. Lu, and B. Wu, "Multilayered Fault Detection and Localization With Transformer for Microservice Systems," *IEEE Trans. Reliab.*, vol. 73, no. 3, pp. 1502–1515, 2024, doi: 10.1109/TR.2024.3356717.

[11] Iman Kohyarnejadfard, Daniel Aloise, Seyed Vahid Azhari & Michel R. Dagenais, "Anomaly detection in microservice environments using distributed tracing data analysis and NLP," *J. Cloud Comput.*, vol. 11, 2022, [Online]. Available: https://link.springer.com/article/10.1186/s13677-022-00296-4

[12] Chenyu Zhao, Minghua Ma, Zhenyu Zhong, Shenglin Zhang, Zhiyuan Tan, Xiao Xiong, LuLu Yu, Jiayi Feng, Yongqian Sun, "Robust Multimodal Failure Detection for Microservice Systems," *arXiv:2305.18985*, 2023, [Online]. Available: https://arxiv.org/abs/2305.18985

[13] C. Zhang, "DeepTraLog: Trace-Log Combined Microservice Anomaly Detection through Graph-based Deep Learning," *2022 IEEE/ACM 44th Int. Conf. Softw. Eng. (ICSE), Pittsburgh, PA, US*, 2022, doi: 10.1145/3510003.3510180.

[14] Ghaith Dkmak, Baris Can, "AI-Driven Anomaly Detection in Cloud-Native Microservices: The Night's Watch Algorithm," *Appl. Sci.*, vol. 15, no. 23, p. 12762, 2025, doi: 10.3390/app152312762.

[15] Ashwin Chavan, "Managing Scalability and Cost in Microservices ArchitectureBalancing Infinite Scalability with Financial Constraints," *J. Med. Healthc.*, 2023, doi: 10.47363/JMHC/2023(5)E102.

[16] Preyashi Agarwal, J. Lakshmi, "Cost Aware Resource Sizing and Scaling of Microservices," *ACM Int. Conf. Proceeding Ser.*, 2019, [Online]. Available: https://dl.acm.org/doi/10.1145/3361821.3361823

[17] M. Khanahmadi, A. Shameli-Sendi, M. Jabbarifar, Q. Fournier, and M. Dagenais, "Detection of microservice-based software anomalies based on OpenTracing in cloud," *Softw. - Pract. Exp.*, vol. 53, no. 8, pp. 1681–1699, Aug. 2023, doi: 10.1002/spe.3208.

[18] L. Wu, J. Tordsson, E. Elmroth, and O. Kao, "MicroRCA: Root Cause Localization of Performance Issues in Microservices," *Proc. IEEE/IFIP Netw. Oper. Manag. Symp. 2020 Manag. Age Softwarization Artif. Intell. NOMS 2020*, Apr. 2020, doi: 10.1109/NOMS47738.2020.9110353.

[19] "Automated Root Cause Analysis Using Artificial Intelligence in Microservices-Oriented DevOps Frameworks | Request PDF." Accessed: Mar. 24, 2026. [Online]. Available: https://www.researchgate.net/publication/394081700_Automated_Root_Cause_Analysis_Using_Artificial_Intelligence_in_Microservices-Oriented_DevOps_Frameworks

[20] Ruyue Xin, Peng Chen, "CausalRCA: Causal inference based precise fine-grained root cause localization for microservice applications," *J. Syst. Softw.*, vol. 203, p. 111724, 2023, doi: https://doi.org/10.1016/j.jss.2023.111724.

[21] Cheryl Lee, Tianyi Yang, Zhuangbin Chen, Yuxin Su, Michael R. Lyu, "Eadro: An End-to-End Troubleshooting Framework for Microservices on Multi-source Data," *arXiv:2302.05092*, 2023, [Online]. Available: https://arxiv.org/abs/2302.05092

[22] "Integrated Real-Time Observability Framework for Transactional Microservices." Accessed: Apr. 20, 2026. [Online]. Available: https://www.researchgate.net/publication/395232600_Integrated_Real-Time_Observability_Framework_for_Transactional_Microservices

[23] Miguel De la Cruz Cabello, Tiago Prince Sales, "AIOps for log anomaly detection in the era of LLMs: A systematic literature review," *Intell. Syst. with Appl.*, vol. 28, p. 200608, 2025, doi: https://doi.org/10.1016/j.iswa.2025.200608.

[24] Eduardo Teixeira Leite, "Cloud Cost Optimization: Strategies for Efficient Resource Management and Infrastructure Cost Reduction in Modern Cloud Environments," *Rev. Sist.*, vol. 14, no. 1, 2023, doi: 10.56238/rcsv14n1-003.

[25] S. Mustafa, K. Bilal, S. U. R. Malik and S. A. Madani, "SLA-Aware Energy Efficient Resource Management for Cloud Environments," *IEEE Access*, vol. 6, pp. 15004–15020, 2018, doi: 10.1109/ACCESS.2018.2808320.

[26] Tariq Al-Omari, "Context-Aware Anomaly Detection in Microservices Using GCN-Encoded Trace Graphs and LSTM-AE Metrics with Local and Global Embeddings," *Eng. Technol. Appl. Sci. Res.*, vol. 15, no. 6, pp. 29277–29283, 2025, doi: 10.48084/etasr.13590.

[27] Q. Li *et al.*, "RAMBO: Resource Allocation for Microservices Using Bayesian Optimization," *IEEE Comput. Archit. Lett.*, vol. 20, no. 1, pp. 46–49, Jan. 2021, doi: 10.1109/LCA.2021.3066142.

[28] "(PDF) Monitoring and Observability for Cloud-Native Applications." Accessed: Apr. 20, 2026. [Online]. Available: https://www.researchgate.net/publication/399515451_Monitoring_and_Observability_for_Cloud-Native_Applications

[29] Avinash Mysore Geethananda, "Finops In Multi-Cloud AI Environments: Financial Governance Strategies For Complex Computational Workloads," *J. Int. Cris. Risk Commun. Res.*, 2025, doi: 10.63278/jicrcr.vi.3460.