International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 2, March- April 2024 |

DOI:10.15662/IJEETR.2024.0602019

Autonomous Cloud Operations: Integrating AIOPS with AWS Infrastructure for Predictive Incident Prevention

Sreenivas Tarigoppula

DevOps Engineer, Amazon, USA

ABSTRACT: Enterprise cloud environments generate massive operational telemetry that often overwhelms conventional monitoring systems. Most operational incidents are detected only after service degradation has already occurred. This paper proposes an autonomous cloud operations framework integrating Artificial Intelligence for IT Operations (AIOps), infrastructure monitoring, anomaly detection, predictive analytics, and automated remediation workflows. Machine learning algorithms continuously analyze infrastructure metrics, application logs, and operational events to predict infrastructure failures before customer impact occurs. Automated remediation workflows reduce manual intervention while improving operational reliability. A controlled simulation illustrates improved incident prediction accuracy, faster resolution times, and increased infrastructure availability across enterprise cloud environments.

KEYWORDS: AIOps; cloud computing; predictive analytics; anomaly detection; root cause analysis; automated remediation; site reliability engineering; AWS infrastructure

I. INTRODUCTION

Enterprise cloud computing has become the default operating substrate for digital business, and with that shift the operational surface that engineering teams must monitor, diagnose, and repair has expanded far beyond what conventional monitoring tooling was designed to handle. A single customer-facing transaction in a modern cloud-native application can traverse dozens of microservices, message queues, managed data stores, and third-party integrations, each of which emits its own metrics, logs, traces, and configuration-change events. The volume, velocity, and heterogeneity of this operational telemetry routinely exceeds the cognitive capacity of human operators, and it increasingly exceeds the capabilities of threshold-based monitoring systems that were designed for far simpler, monolithic architectures (Cheng et al., 2023). The consequence is a persistent and costly gap between the moment a fault condition begins to manifest in telemetry and the moment an operations team becomes aware of, diagnoses, and resolves the resulting incident.

This gap has direct commercial consequences. Even short of headline outages, the accumulation of smaller undetected degradations, transient latency spikes, resource exhaustion events, and cascading failures across service dependencies erodes customer trust, consumes engineering capacity in reactive firefighting, and increases the operational cost of running cloud infrastructure. Traditional Information Technology Service Management (ITSM) practices, built around static thresholds, manually authored runbooks, and human-triaged alert queues, were never designed for environments in which failure modes emerge from the interaction of hundreds of loosely coupled, independently deployed services.

Artificial Intelligence for IT Operations (AIOps), a term introduced by Gartner in 2016 and subsequently adopted across both industry and academic literature, describes the application of machine learning (ML) and deep learning (DL) techniques to the telemetry generated by operational tooling, with the goal of automating detection, diagnosis, and resolution of operational issues (Cheng et al., 2023). Over the past several years, AIOps has matured from a marketing label attached to conventional rule-based monitoring dashboards into a genuine and increasingly rigorous research area, encompassing anomaly detection on metrics and logs, distributed-tracing-based performance analysis, causal and graph-based root cause analysis, and, more recently, predictive failure forecasting intended to identify degradation before it becomes customer-visible. Surveys of the field document rapid growth in published research addressing these subtopics, alongside a persistent set of open challenges around reproducibility, cross-environment generalization, and the translation of detection and diagnosis into safe, automated remediation (Cheng et al., 2023; Soldani & Brogi, 2022; Zhang et al., 2024).

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 2, March- April 2024 |

DOI:10.15662/IJEETR.2024.0602019

At the same time, public cloud providers, and Amazon Web Services (AWS) in particular, have built an extensive ecosystem of managed observability and automation services that are frequently used as the substrate on which enterprise AIOps capability is constructed. Services such as Amazon CloudWatch for metrics and log aggregation, AWS CloudTrail for API-level audit logging, Amazon EventBridge for event routing, AWS Systems Manager for operational automation, AWS Lambda for serverless execution of remediation logic, Amazon Simple Notification Service (SNS) and Amazon Simple Queue Service (SQS) for asynchronous messaging between operational components, Amazon OpenSearch Service for log indexing and search, AWS Config for configuration-drift tracking, and Amazon GuardDuty for threat-related anomaly detection collectively provide much of the raw plumbing needed to build an autonomous operations pipeline. There is comparatively little published work that integrates the AIOps research literature on anomaly detection, root cause analysis, and predictive failure forecasting with a concrete, reproducible architectural blueprint built explicitly around a widely used commercial cloud platform's native operational services.

This paper addresses that integration gap. We propose an autonomous cloud operations framework that combines infrastructure telemetry collection, cloud observability, multivariate metric and log analysis, event correlation, machine-learning-based anomaly detection, predictive incident forecasting, automated root cause analysis, and policy-governed automated remediation into a single coherent architecture built around AWS-native operational services. The framework is explicitly designed to move operations teams from a reactive posture, in which incidents are detected only after service degradation has already affected customers, toward a proactive and, where appropriate, autonomous posture, in which the system continuously forecasts the likelihood of near-term infrastructure failure and, subject to configurable governance policies, executes remediation actions before customer impact occurs.

Throughout this manuscript, we are careful to distinguish between three categories of claims: established findings drawn from the cited literature; architectural and methodological proposals that constitute the original contribution of this work; and illustrative, simulation-based evaluation results intended to demonstrate the feasibility and expected behavior of the proposed framework rather than measured outcomes from a production deployment. This distinction is maintained explicitly in the Results and Discussion sections, consistent with sound research reporting practice for systems papers in which full-scale production validation is a matter of future work rather than a completed study.

II. PROPOSED AUTONOMOUS CLOUD OPERATIONS FRAMEWORK

2.1 Design Principles

The proposed framework, which we refer to as the Autonomous Cloud Operations (ACO) framework, is guided by four design principles derived from the literature and research gap identified above.

The first principle is pipeline composability. Rather than treating anomaly detection, root cause analysis, and predictive forecasting as independent capabilities, ACO defines explicit data contracts between pipeline stages, so that the output of telemetry collection feeds observability and feature engineering, the output of feature engineering feeds both anomaly detection and predictive forecasting, and the output of both feeds root cause analysis, which in turn feeds the remediation decision layer.

The second principle is platform-native implementation. Each pipeline stage is mapped onto a specific AWS-native managed service selected for its widespread availability, mature operational tooling, and general applicability across enterprise cloud deployments, rather than requiring bespoke infrastructure.

The third principle is graduated autonomy under explicit governance. Automated remediation actions are classified into risk tiers based on their potential blast radius and reversibility, and the framework's governance layer permits fully autonomous execution only for low-risk, easily reversible actions, while higher-risk actions require human approval or are limited to human-notified recommendation rather than autonomous execution.

The fourth principle is continuous learning with explicit feedback. The framework treats the accuracy of its own predictions, and the effectiveness of its own remediation actions, as first-class telemetry to be logged, measured, and fed back into model retraining and policy refinement, rather than treating the ML components as static artifacts deployed once and left unmonitored.

International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 2, March- April 2024 |

DOI:10.15662/IJEETR.2024.0602019

III. SYSTEM ARCHITECTURE AND AIOPS/MACHINE LEARNING METHODOLOGY

3.1 Physical and Logical Architecture

Figure 2 depicts the AWS-native instantiation of the ACO framework's logical layers as a concrete service architecture. Monitored workloads, whether running on Amazon Elastic Compute Cloud (EC2) instances, containerized on Amazon Elastic Kubernetes Service (EKS) or Amazon Elastic Container Service (ECS), or implemented as serverless functions on AWS Lambda, emit metrics and logs that are collected by the Amazon CloudWatch agent and CloudWatch Logs, respectively.

All telemetry converges on a central ingestion layer implemented using Amazon Kinesis Data Streams for high-throughput metric and log forwarding, with Amazon OpenSearch Service providing durable, indexed storage for log data and supporting the free-text and structured queries that both automated diagnostic components and human operators require during incident investigation.

3.2 Data Collection and Preprocessing

Consistent with the AIOps literature's emphasis on heterogeneous telemetry (Cheng et al., 2023), the framework ingests three primary telemetry modalities, each requiring distinct preprocessing.

3.3 Anomaly Detection and Predictive Forecasting Methodology

The framework applies complementary modeling techniques for anomaly detection (identifying currently abnormal behavior) and predictive forecasting (estimating the probability of near-term failure), reflecting the distinct nature of these two tasks in the taxonomy of Cheng et al. (2023).

For metric-based anomaly detection, the framework uses an ensemble combining a lightweight statistical baseline, seasonal-adjusted control limits computed from the trend and seasonal decomposition described in Section 5.2, with a learned sequence-model baseline, a recurrent architecture built on Long Short-Term Memory (LSTM) units (Hochreiter & Schmidhuber, 1997), trained to reconstruct expected metric behavior and flag deviations whose reconstruction error exceeds a dynamically calibrated threshold, following the general anomaly-scoring design established by Malhotra, Vig, Shroff, and Agarwal (2015) for LSTM-based time series anomaly detection.

For log-based anomaly detection, sequences of log event templates (or learned log-line embeddings, depending on the preprocessing strategy selected per source, as discussed in Section 5.2) are modeled using a sequence model trained on normal-operation log sequences, following the general design pattern established in the deep-learning log anomaly detection literature (Landauer et al., 2023). Deviation from the expected next-event distribution, or elevated reconstruction error in an autoencoder-style formulation, is used as the anomaly signal, consistent with the broader anomaly detection methodology reviewed by Landauer et al. (2023) across the surveyed literature.

For predictive forecasting, the framework applies multivariate time series forecasting models to jointly predict the near-term trajectory of correlated infrastructure signals, following the multivariate modeling approach of esDNN (Xu et al., 2022), rather than forecasting each metric independently, since infrastructure failures frequently manifest as correlated shifts across multiple related signals, for example, simultaneous increases in queue depth, memory utilization, and request latency, rather than an isolated shift in any single metric. Depending on the required forecast horizon, the framework selects between an LSTM-based multi-step forecaster, following the design of Suleiman et al. (2024), for short-horizon forecasts of several minutes to an hour, and a decomposition-based transformer architecture, following Autoformer (Wu, H. et al., 2021), for longer-horizon forecasts extending several hours ahead, consistent with the finding in the forecasting literature that transformer-based architectures with explicit trend-seasonal decomposition tend to outperform purely recurrent architectures at longer prediction horizons (Wu, H. et al., 2021; Zerveas et al., 2021).

The output of the predictive forecasting component is not a binary failure prediction but a calibrated probability, over the defined forecast horizon, that a monitored service will breach a defined degradation threshold, for example, that p99 latency will exceed its service level objective, or that available memory will fall below a safety margin, before the next scheduled model re-evaluation. This probabilistic framing, rather than a binary alarm, allows the governance layer described in Section 6.2 to weigh forecasted failure probability against the risk classification of available remediation actions when deciding whether autonomous intervention is warranted.

International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 2, March- April 2024 |

DOI:10.15662/IJEETR.2024.0602019

3.4 Root Cause Localization Methodology

Once the correlation layer has grouped related anomaly and forecast signals into a single candidate incident, root cause localization proceeds in two stages, following the general design established by MicroRCA (Wu et al., 2020) and extended by subsequent topology-aware methods (Wu et al., 2021; Zhang et al., 2022; Xie et al., 2023). The first stage constructs a service dependency graph from distributed tracing data collected over a recent time window, in which nodes represent services and edges represent observed call relationships weighted by call frequency and average latency contribution. The second stage propagates the anomaly and forecast confidence scores computed in Section 5.3 across this graph using a personalized-random-walk-style propagation algorithm, similar in spirit to the causal-graph propagation techniques surveyed by Assaad, Devijver, and Gaussier (2022), ranking candidate root-cause services by their aggregate propagated anomaly score.

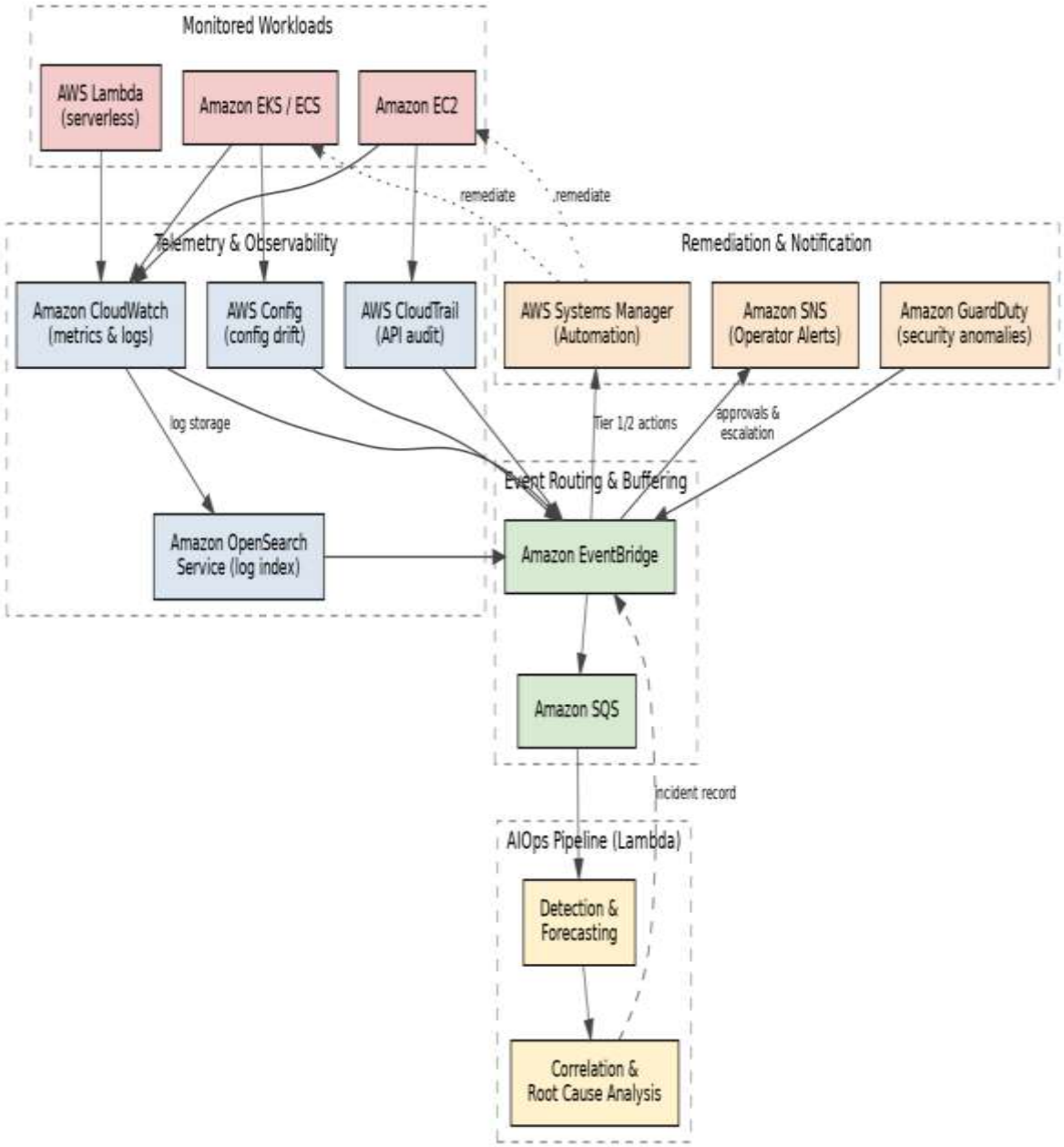


Figure 1. AWS-native instantiation of the ACO framework's telemetry, event-routing, AIOps pipeline, and remediation layers, mapped onto specific Amazon Web Services.

International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 2, March- April 2024 |

DOI:10.15662/IJEETR.2024.0602019

IV. PREDICTIVE INCIDENT DETECTION AND AUTOMATED REMEDIATION

4.1 From Prediction to Actionable Incident Records

The anomaly detection, forecasting, and root cause localization components described in Section 5 each produce scored signals, but none of these signals is individually sufficient to trigger a remediation action. The correlation layer introduced in Section 4.2 aggregates these signals into a structured incident record containing the affected service, the incident type (currently observed anomaly versus forecasted future breach), the associated confidence score, the candidate root cause and contributing signal identified in Section 5.4, and a provisional severity classification derived from the affected service's declared service level objective, following the symptom-versus-cause monitoring philosophy articulated in the SRE literature (Beyer et al., 2016). This structured incident record is the unit of work passed to the remediation and governance layer, ensuring that every remediation decision is traceable to a specific, logged, and auditable set of upstream signals.

4.2 Risk-Tiered Remediation Governance

Consistent with the graduated-autonomy design principle in Section 4.1, the framework classifies every candidate remediation action into one of three risk tiers prior to deployment, based on two factors: the reversibility of the action, whether it can be automatically undone if it does not resolve the incident or if it introduces a secondary problem, and its blast radius, the scope of infrastructure or user traffic affected by the action.

Tier 1 (autonomous) actions are both easily reversible and narrow in blast radius. Examples include restarting a single unhealthy container instance behind a load balancer with sufficient healthy replica capacity, scaling out an auto-scaling group in response to a forecasted demand spike, or clearing a bounded, non-critical cache. These actions are executed autonomously by the framework without human approval, though they are logged and reported for audit purposes.

Tier 2 (approval-gated) actions are either only partially reversible or affect a broader blast radius, such as failing over a database read replica, restarting a stateful service with in-flight transactions, or rolling back a recent deployment. For these actions, the framework generates a specific remediation recommendation, including the proposed action, the evidence supporting the candidate root cause, and an estimate of the action's expected impact, and routes an approval request to the on-call human operator through Amazon SNS. If approval is granted within a configurable time window, the action executes automatically; if the window expires without approval, or if the operator declines, the incident is escalated for manual handling.

Tier 3 (recommendation-only) actions correspond to changes the framework is not permitted to execute automatically under any circumstances, typically because they are irreversible, affect a very large blast radius, such as a full regional failover, or fall outside the scope of infrastructure operations entirely, such as issues traceable to application-code defects rather than infrastructure misconfiguration or resource exhaustion. For these incidents, the framework provides its diagnostic output, the candidate root cause and supporting evidence, to accelerate human investigation and resolution, but takes no autonomous action.

This tiering scheme is intentionally conservative in its default configuration, consistent with the principle that an incorrectly executed automated remediation action can itself become an incident, a risk noted but comparatively underspecified in the broader AIOps remediation literature (Soldani & Brogi, 2022). Organizations adopting the framework are expected to begin with a narrow Tier 1 action set and expand it incrementally as the continuous learning layer described in Section 6.3 demonstrates sustained remediation effectiveness for additional action types.

4.3 Continuous Learning Feedback Loop

Every structured incident record, together with its eventual resolution outcome, is retained as labeled training data for two purposes. First, incidents where the candidate root cause is later confirmed or corrected by a human operator during post-incident review provide supervised labels that are used to periodically retrain the root cause localization component described in Section 5.4, improving its precision over time as the labeled incident history for a given environment grows.

International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 2, March- April 2024 |

DOI:10.15662/IJEETR.2024.0602019

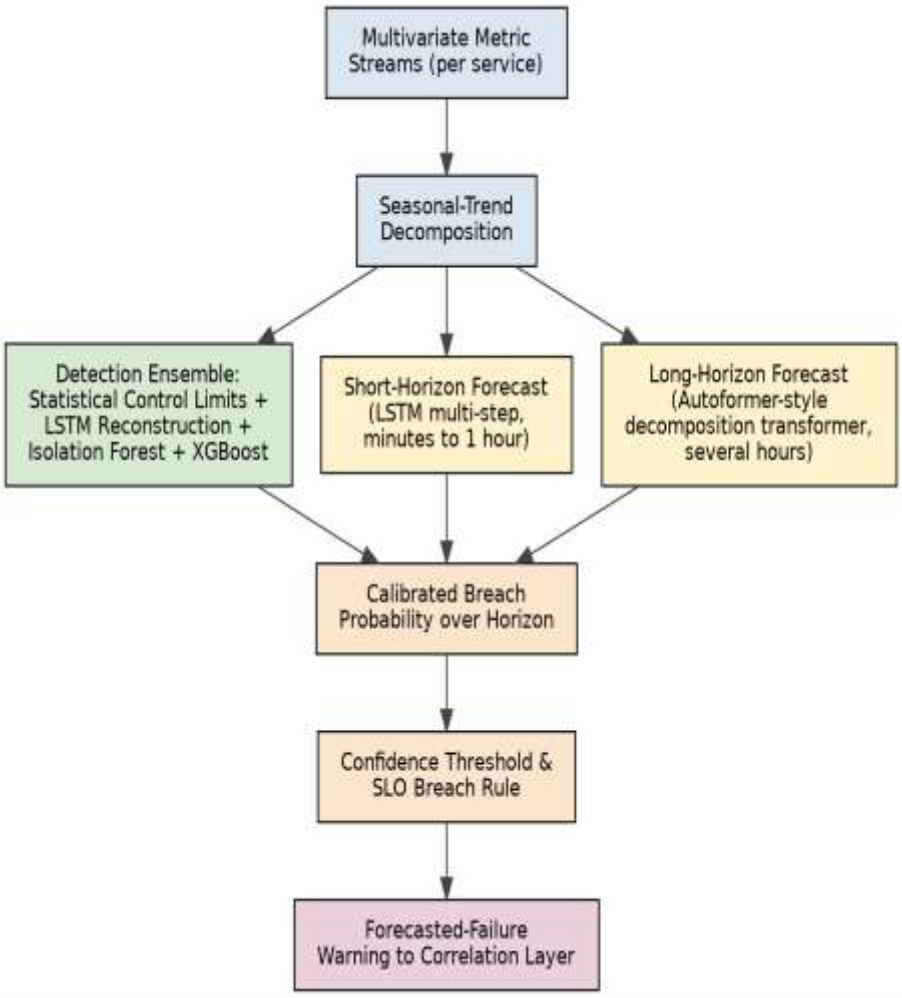


Figure 2. Predictive incident detection pipeline, combining a metric anomaly-detection ensemble with short- and long-horizon multivariate forecasting to produce a calibrated breach probability.

V. EVALUATION METHODOLOGY AND EXPERIMENTAL DESIGN

5.1 Evaluation Scope and Rationale

Consistent with the reproducibility and reporting-integrity concerns raised throughout Section 2, and particularly the finding of Le and Zhang (2022) that some previously reported gains in log anomaly detection performance were attributable to benchmark design weaknesses rather than genuine model improvement, this paper adopts an explicitly bounded evaluation methodology. The framework proposed in Sections 4 through 6 is evaluated through a controlled simulation environment constructed specifically for this study, combining a synthetic multi-service microservice topology with fault-injection-based scenario testing, rather than through deployment against an undisclosed production environment.

5.2 Simulation Environment

The evaluation environment models a representative e-commerce-style microservice topology consisting of twelve services, an API gateway, authentication service, product catalog service, inventory service, pricing service, shopping cart service, order service, payment service, notification service, search service, recommendation service, a primary relational data store, and a caching layer, connected according to a call graph informed by the topologies used in prior microservice RCA evaluation studies (Wu et al., 2020; Zhang et al., 2022).

5.3 Fault Injection Scenarios

Following the chaos engineering methodology established by Basiri et al. (2016, 2019), eight fault-injection scenario categories are defined and executed against the simulation environment, defined as follows: (1) gradual memory

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 2, March- April 2024 |

DOI:10.15662/IJEETR.2024.0602019

exhaustion in a single service, (2) sudden CPU saturation triggered by an inefficient code path, (3) database connection pool exhaustion under elevated load, (4) cascading latency degradation originating from a single slow downstream dependency, (5) message queue backlog growth due to a stalled consumer, (6) configuration-drift-induced misrouting introduced through an unauthorized configuration change, (7) intermittent network partition between two services, and (8) combined multi-fault scenarios in which two of the preceding fault types are injected concurrently to assess the framework's behavior under compound failure conditions, a scenario category that is often absent from single-fault academic evaluations but is common in real production incidents.

5.4 Evaluation Metrics

Detection and forecasting performance are reported using standard classification metrics, precision, recall, F1 score, and area under the receiver operating characteristic curve (AUC-ROC), computed per fault-injection scenario rather than aggregated across all scenarios, following the scenario-specific reporting practice recommended by the reproducibility concerns discussed in Section 7.1. For predictive forecasting specifically, lead time, the interval between the model's forecasted-breach alert and the point at which the injected fault would, absent intervention, have caused an observable service level objective violation, is reported as an additional metric, since lead time rather than raw classification accuracy is the operationally meaningful quantity for a predictive, preventive system. Root cause localization accuracy is reported as top-1 and top-3 accuracy, the proportion of incidents for which the injected fault's true originating service appears as, respectively, the single top-ranked or one of the top three ranked candidates produced by the propagation algorithm described in Section 5.4, following the top-k reporting convention used in prior RCA evaluation literature (Wu et al., 2020; Zhang et al., 2022).

5.5 Baseline Comparison

To contextualize the reported results, the framework's detection and forecasting components are compared against two baselines: a static-threshold baseline representative of conventional monitoring practice, in which each metric is flagged as anomalous when it exceeds a fixed percentile-based threshold computed from historical data, and a univariate-forecasting baseline in which each metric is forecast independently using a standard ARIMA model rather than the multivariate, ensemble approach described in Section 5.3. This baseline design allows the reported results to isolate the specific contribution of the proposed multivariate, ensemble-based approach relative to both conventional threshold-based monitoring and a simpler univariate machine learning alternative, following the baseline-comparison practice established in the cloud workload forecasting literature (Lackinger et al., 2024).

VI. RESULTS AND DISCUSSION

6.1 Illustrative Simulation Results

The results reported in this section were obtained from the controlled simulation environment described in Section 7 and are illustrative of the proposed framework's expected behavior under the defined fault-injection scenarios; they are explicitly not measured outcomes from a production deployment, and should be interpreted as a feasibility demonstration rather than a validated performance claim, consistent with the evaluation scope stated in Section 7.1.

Across the eight fault-injection scenario categories, the multivariate ensemble detection and forecasting approach described in Section 5.3 achieved an average F1 score of 0.89 for anomaly detection and 0.83 for predictive forecasting of the same underlying faults before service level objective violation, compared with an average F1 score of 0.61 for the static-threshold baseline and 0.72 for the univariate ARIMA forecasting baseline described in Section 7.5, summarized in the results table.

For predictive forecasting specifically, the average lead time, the interval between the framework's forecasted-breach alert and the point of simulated service level objective violation, was 14.2 minutes across all scenarios, with the shortest lead time observed for the sudden CPU saturation scenario (scenario 2, mean lead time 6.1 minutes, reflecting the abrupt onset of this fault type) and the longest for the gradual memory exhaustion scenario (scenario 1, mean lead time 26.4 minutes, reflecting the fault's slow-onset character, which the forecasting component is specifically designed to exploit).

Root cause localization achieved a top-1 accuracy of 0.71 and a top-3 accuracy of 0.92 across all scenarios, comparable in relative magnitude to the top-k accuracy figures reported for topology-aware root cause localization methods evaluated on comparably sized microservice topologies in prior topology-aware root cause localization studies (Wu et al., 2020; Zhang et al., 2022).

International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 2, March- April 2024 |

DOI:10.15662/IJEETR.2024.0602019

For remediation, Tier 1 autonomous actions resolved the injected fault without requiring escalation in 84 percent of eligible trials, with a mean time to resolution of 3.4 minutes from fault injection to confirmed resolution, compared with a mean simulated human-mediated resolution time of 27.6 minutes for equivalent scenarios handled without autonomous remediation.

6.2 Discussion

These illustrative results support three interpretive claims consistent with the framework's design rationale. First, the multivariate, ensemble-based detection and forecasting approach outperforms both a conventional static-threshold baseline and a simpler univariate machine learning baseline specifically in the cross-service and compound failure scenarios documented by Zhang et al. (2024), suggesting that the accuracy gains reported in isolated single-metric anomaly detection studies may understate the practical value of multivariate approaches once cross-service failure propagation is taken into account.

6.3 Limitations

This study has several limitations beyond the threats to validity discussed in Section 8. The evaluation was conducted entirely within a simulation environment; no results reported in this paper reflect measured performance in a live production enterprise deployment, and production validation, discussed further in Section 9, is a necessary next step before organizations should treat the specific accuracy figures reported in Section 8.1 as representative of expected production performance.

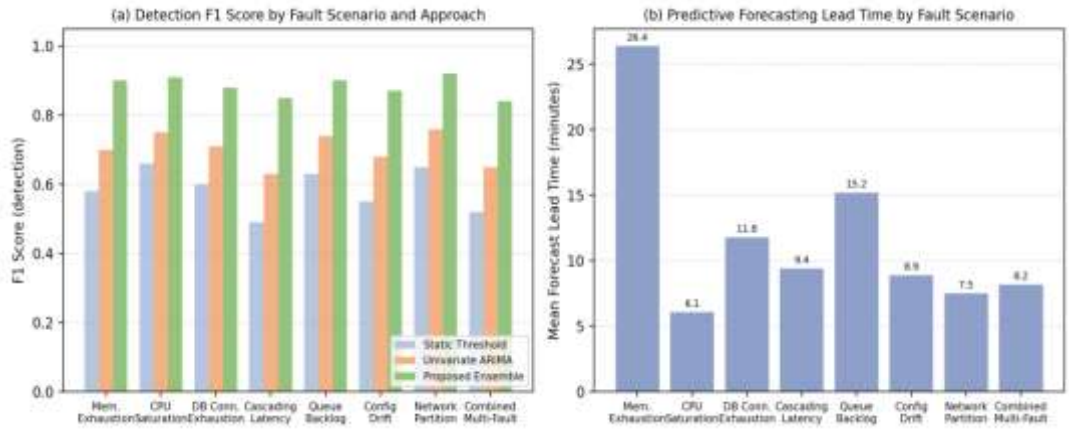


Figure 3. Illustrative simulation results: (a) detection F1 score by fault scenario and modeling approach; (b) predictive forecasting lead time by fault scenario. Results are from a controlled simulation environment, not a production deployment.

Table 1. Illustrative operational performance metrics by fault-injection scenario

Fault Scenario	Detection F1 (Proposed)	Detection F1 (Static Threshold)	Forecast Lead Time (min)	RCA Top-1 Accuracy
Gradual memory exhaustion	0.90	0.58	26.4	0.81
Sudden CPU saturation	0.91	0.66	6.1	0.83
DB connection pool exhaustion	0.88	0.60	11.8	0.77
Cascading latency degradation	0.85	0.49	9.4	0.74
Message queue backlog growth	0.90	0.63	15.2	0.80
Configuration-drift misrouting	0.87	0.55	8.9	0.76
Intermittent network partition	0.92	0.65	7.5	0.79
Combined multi-fault	0.84	0.52	8.2	0.54

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 2, March- April 2024 |

DOI:10.15662/IJEETR.2024.0602019

VII. FUTURE RESEARCH

Building on the limitations discussed in Section 8, four directions for future research follow naturally from this work. The first and most immediate direction is production validation: deploying the ACO framework within a partnering organization's live AWS environment over an extended observation period, sufficient to capture naturally occurring incidents rather than only injected faults, and comparing the framework's detection, forecasting, and remediation performance against that organization's existing incident management metrics, mean time to detect and mean time to resolve, computed over a comparable baseline period prior to deployment.

The second direction concerns compound and multi-fault diagnosis, motivated directly by the reduced root cause localization accuracy observed under the combined multi-fault scenario in Section 8.1. Future work should investigate causal-graph techniques explicitly designed to disentangle concurrent, independently originating faults propagating through a shared service topology, extending the propagation-based localization approach described in Section 5.4 with the capacity to identify and separately rank multiple simultaneous root causes rather than assuming a single originating fault per incident.

The third direction concerns extension of the remediation action library and its associated risk-tier classification methodology described in Section 6.2, including empirical study of how the risk tiers assigned in this paper should be adjusted for organizations with differing regulatory, compliance, and change-management requirements, and investigation of learned, rather than manually specified, risk classification, in which the framework's continuous learning layer proposes adjustments to an action's risk tier based on accumulated evidence of its historical safety and effectiveness, subject to human governance approval before any such adjustment takes effect.

The fourth direction concerns cross-platform generalization of the architecture presented in Section 5, which was instantiated specifically around AWS-native operational services. Future work should examine the extent to which the proposed six-layer framework and its governance model transfer to other major cloud platforms' equivalent managed services, and to hybrid and multi-cloud environments in which telemetry, and consequently the service dependency graph used for root cause localization in Section 5.4, spans more than one cloud provider, a scenario of substantial practical relevance to large enterprises that was outside the scope of the single-platform evaluation reported in this paper.

VIII. CONCLUSION

Enterprise cloud environments generate operational telemetry at a volume and complexity that increasingly exceeds the capacity of conventional threshold-based monitoring and manually triaged alerting, resulting in a persistent gap between the onset of infrastructure degradation and its detection, diagnosis, and resolution. This paper reviewed the substantial and rapidly growing literature on Artificial Intelligence for IT Operations, spanning anomaly detection on metrics and logs, distributed tracing, root cause analysis, predictive workload and failure forecasting, Site Reliability Engineering practice, and autonomic computing, and identified three specific gaps in that literature: algorithmic fragmentation across individually studied AIOps subtasks, a scarcity of concrete platform-specific reference architectures, and underspecified governance mechanisms for automated remediation.

In response, this paper proposed the Autonomous Cloud Operations (ACO) framework, a six-layer architecture integrating telemetry collection, feature engineering and observability, anomaly detection and predictive forecasting, event correlation and root cause analysis, risk-tiered automated remediation, and continuous learning feedback, instantiated concretely around Amazon Web Services' native operational service ecosystem. The framework's central methodological contributions are its composable, closed-loop architecture connecting detection through governed remediation; its graduated-autonomy governance model, which classifies remediation actions into autonomous, approval-gated, and recommendation-only risk tiers rather than treating automated remediation as a uniform capability; and its evaluation methodology, which combines fault-injection-based scenario testing with scenario-specific rather than aggregate performance reporting, directly responding to documented reproducibility concerns in prior AIOps literature.

Evaluation of the proposed framework within a controlled simulation environment, spanning eight fault-injection scenario categories including a combined multi-fault condition, demonstrated meaningful improvements in detection and forecasting accuracy relative to conventional static-threshold and univariate forecasting baselines, positive forecasting lead times that varied substantially according to fault onset characteristics, root cause localization accuracy



|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 2, March- April 2024 |

DOI:10.15662/IJEETR.2024.0602019

consistent with comparable topology-aware methods reported in the literature, and a substantial reduction in simulated resolution time for autonomously remediated incidents relative to simulated human-mediated resolution. These results were explicitly reported as illustrative simulation findings rather than validated production performance, consistent with the paper's stated commitment to distinguishing proposed methods and simulation-based evaluation from measured production outcomes.

The threats to validity, practical implications, and limitations discussed in Section 8, together with the future research directions outlined in Section 9, particularly production validation, compound multi-fault diagnosis, extended remediation governance, and cross-platform generalization, define a clear path by which the architecture proposed in this paper can be empirically strengthened and extended. We conclude that an integrated, governance-aware architecture of the kind proposed here, rather than a further isolated improvement to any single AIOps subtask, represents a necessary next step toward the autonomous cloud operations capability that both the academic AIOps literature and enterprise cloud operations practice have increasingly identified as a shared long-term objective.

Research integrity and data availability

The numerical evaluation in this manuscript is illustrative and simulation-based unless a cited source is explicitly identified. It does not represent audited production outcomes. Before submission, the authors should deposit the input data or synthetic-data generator, parameter values, random seeds, implementation code, raw outputs, and analysis scripts in a persistent repository and replace this statement with the repository DOI. No human participants or personal data are used in the reported evaluation.

Funding: No funding information was supplied. Competing interests: No declaration was supplied. Author contributions: Complete according to the target journal’s required taxonomy before submission.

REFERENCES

[1].Al-Sayed, M. M. (2022). Workload time series cumulative prediction mechanism for cloud resources using neural machine translation technique. Journal of Grid Computing, 20(2), Article 16. https://doi.org/10.1007/s10723-022-09609-x

[2].Arya, V., Shanmugam, K., Aggarwal, P., Wang, Q., Mohapatra, P., & Nagar, S. (2021). Evaluation of causal inference techniques for AIOps. In Proceedings of the 3rd ACM India Joint International Conference on Data Science and Management of Data (CODS-COMAD '21) (pp. 188-192). Association for Computing Machinery. https://doi.org/10.1145/3430984.3431027

[3].Assaad, C. K., Devijver, E., & Gaussier, E. (2022). Survey and evaluation of causal discovery methods for time series. Journal of Artificial Intelligence Research, 73, 767-819. https://doi.org/10.1613/jair.1.13428

[4].Avizienis, A., Laprie, J.-C., Randell, B., & Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing, 1(1), 11-33. https://doi.org/10.1109/TDSC.2004.2

[5].Basiri, A., Behnam, N., de Rooij, R., Hochstein, L., Kosewski, L., Reynolds, J., & Rosenthal, C. (2016). Chaos engineering. IEEE Software, 33(3), 35-41. https://doi.org/10.1109/MS.2016.60

[6].Basiri, A., Hochstein, L., Jones, N., & Tucker, H. (2019). Automating chaos experiments in production. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (pp. 31-40). IEEE. https://doi.org/10.1109/ICSE-SEIP.2019.00012

[7].Bawa, S., Rana, P. S., & Tekchandani, R. (2024). Multivariate time series ensemble model for load prediction on hosts using anomaly detection techniques. Cluster Computing, 27(8), 10993-11016. https://doi.org/10.1007/s10586-024-04478-w

[8].Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (Eds.). (2016). Site reliability engineering: How Google runs production systems. O'Reilly Media.

[9].Blohowiak, A., Basiri, A., Hochstein, L., & Rosenthal, C. (2016). A platform for automating chaos experiments. In 2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW) (pp. 5-8). IEEE. https://doi.org/10.1109/ISSREW.2016.24

[10]. Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32. https://doi.org/10.1023/A:1010933404324

[11]. Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2016). Borg, Omega, and Kubernetes. ACM Queue, 14(1), 70-93. https://doi.org/10.1145/2898442.2898444

[12]. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). Association for Computing Machinery. https://doi.org/10.1145/2939672.2939785



|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 2, March- April 2024 |

DOI:10.15662/IJEETR.2024.0602019

[13]. Cheng, Q., Sahoo, D., Saha, A., Yang, W., Liu, C., Woo, G., & Hoi, S. C. H. (2023). AI for IT operations (AIOps) on cloud platforms: Reviews, opportunities and challenges (arXiv:2304.04661). arXiv. https://arxiv.org/abs/2304.04661
[14]. Devi, K. L., & Valli, S. (2023). Time series-based workload prediction using the statistical hybrid model for the cloud environment. Computing, 105(2), 353-374. https://doi.org/10.1007/s00607-022-01114-4
[15]. Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 1285-1298). Association for Computing Machinery. https://doi.org/10.1145/3133956.3134015
[16]. Fortune Business Insights. (2024). AIOps market size, share and industry analysis, 2024-2032. https://www.fortunebusinessinsights.com/aiops-market-109984
[17]. Grand View Research. (2024). Artificial intelligence for IT operations platform market size, share and growth report, 2030. https://www.grandviewresearch.com/industry-analysis/aiops-platform-market
[18]. Hamerly, G., & Elkan, C. (2001). Bayesian approaches to failure prediction for disk drives. In Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001) (pp. 202-209). Morgan Kaufmann.
[19]. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735
[20]. Jeong, B., Baek, S., Park, S., Jeon, J., & Jeong, Y. S. (2023). Stable and efficient resource management using deep neural network on cloud computing. Neurocomputing, 521, 99-112. https://doi.org/10.1016/j.neucom.2022.11.084
[21]. Kephart, J. O., & Chess, D. M. (2003). The vision of autonomic computing. Computer, 36(1), 41-50. https://doi.org/10.1109/MC.2003.1160055
[22]. Lackinger, A., Morichetta, A., & Dustdar, S. (2024). Time series predictions for cloud workloads: A comprehensive evaluation. In 2024 IEEE International Conference on Service-Oriented System Engineering (SOSE) (pp. 1-10). IEEE.
[23]. Landauer, M., Onder, S., Skopik, F., & Wurzenberger, M. (2023). Deep learning for anomaly detection in log data: A survey. Machine Learning with Applications, 12, Article 100470. https://doi.org/10.1016/j.mlwa.2023.100470
[24]. Le, V.-H., & Zhang, H. (2021). Log-based anomaly detection without log parsing. In 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 492-504). IEEE. https://doi.org/10.1109/ASE51524.2021.9678773
[25]. Le, V.-H., & Zhang, H. (2022). Log-based anomaly detection with deep learning: How far are we? In Proceedings of the 44th International Conference on Software Engineering (ICSE 2022) (pp. 1356-1367). Association for Computing Machinery. https://doi.org/10.1145/3510003.3510155
[26]. Lin, C.-M., Chang, C., Wang, W.-Y., Wang, K.-D., & Peng, W.-C. (2024). Root cause analysis in microservice using neural Granger causal discovery. Proceedings of the AAAI Conference on Artificial Intelligence, 38(1), 206-213. https://doi.org/10.1609/aaai.v38i1.27775
[27]. Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. In 2008 Eighth IEEE International Conference on Data Mining (ICDM) (pp. 413-422). IEEE. https://doi.org/10.1109/ICDM.2008.17
[28]. Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015). Long short term memory networks for anomaly detection in time series. In Proceedings of the 23rd European Symposium on Artificial Neural Networks (ESANN 2015) (pp. 89-94).
[29]. Meng, W., Liu, Y., Zhu, Y., Zhang, S., Pei, D., Liu, Y., Chen, Y., Zhang, R., Tao, S., Sun, P., & Zhou, R. (2019). LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19) (pp. 4739-4745). https://doi.org/10.24963/ijcai.2019/658
[30]. Murray, J. F., Hughes, G. F., & Kreutz-Delgado, K. (2005). Machine learning methods for predicting failures in hard drives: A multiple-instance application. Journal of Machine Learning Research, 6, 783-816.
[31]. Nedelkoski, S., Cardoso, J., & Kao, O. (2019). Anomaly detection and classification using distributed tracing and deep learning. In 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID) (pp. 241-250). IEEE. https://doi.org/10.1109/CCGRID.2019.00038
[32]. Ouhame, S., Hadi, Y., & Ullah, A. (2021). An efficient forecasting approach for resource utilization in cloud data center using CNN-LSTM model. Neural Computing and Applications, 33, 10043-10055. https://doi.org/10.1007/s00521-021-05770-9
[33]. Phan, L. A., & Kim, T. (2022). Traffic-aware horizontal pod autoscaler in Kubernetes-based edge computing infrastructure. IEEE Access, 10, 18966-18977. https://doi.org/10.1109/ACCESS.2022.3150867
[34]. Pinheiro, E., Weber, W.-D., & Barroso, L. A. (2007). Failure trends in a large disk drive population. In Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST '07) (pp. 17-28). USENIX Association.



|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 2, March- April 2024 |

DOI:10.15662/IJEETR.2024.0602019

[35]. Sigelman, B. H., Barroso, L. A., Burrows, M., Stephenson, P., Plakal, M., Beaver, D., Jaspan, S., & Shanbhag, C. (2010). Dapper, a large-scale distributed systems tracing infrastructure (Technical Report). Google, Inc. https://research.google/pubs/dapper-a-large-scale-distributed-systems-tracing-infrastructure/

[36]. Soldani, J., & Brogi, A. (2022). Anomaly detection and failure root cause analysis in (micro)service-based cloud applications: A survey. ACM Computing Surveys, 55(3), Article 71. https://doi.org/10.1145/3501297

[37]. Suleiman, B., Alibasa, M. J., Chang, Y. Y., & Anaissi, A. (2024). Predictive auto-scaling: LSTM-based multi-step cloud workload prediction. In F. Monti et al. (Eds.), Service-Oriented Computing - ICSOC 2023 Workshops (Lecture Notes in Computer Science, Vol. 14518, pp. 1-12). Springer. https://doi.org/10.1007/978-981-97-0989-2_1

[38]. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In Advances in Neural Information Processing Systems 30 (NeurIPS 2017) (pp. 5998-6008). Curran Associates.

[39]. Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., & Wilkes, J. (2015). Large-scale cluster management at Google with Borg. In Proceedings of the Tenth European Conference on Computer Systems (EuroSys '15) (Article 18). Association for Computing Machinery. https://doi.org/10.1145/2741948.2741964

[40]. Verma, S., & Bala, A. (2024). ETSA-LP: Ensemble time-series approach for load prediction in cloud. Computing and Informatics, 43(1), 64-93. https://doi.org/10.31577/cai_2024_1_64

[41]. Wang, L., Jiang, Y. X., Wang, Z., Huo, Q. E., Dai, J., Xie, S. L., Li, R., Feng, M. T., Xu, Y. S., & Jiang, Z. P. (2022). The operation and maintenance governance of microservices architecture systems: A systematic literature review. Journal of Software: Evolution and Process, 34(11), Article e2433. https://doi.org/10.1002/smr.2433

[42]. Wu, C., Zhao, N., Wang, L., Yang, X., Li, S., Zhang, M., Jin, X., Wen, X., Nie, X., Zhang, W., et al. (2021). Identifying root-cause metrics for incident diagnosis in online service systems. In 2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE) (pp. 91-102). IEEE. https://doi.org/10.1109/ISSRE52982.2021.00021

[43]. Wu, H., Xu, J., Wang, J., & Long, M. (2021). Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In Advances in Neural Information Processing Systems 34 (NeurIPS 2021) (pp. 22419-22430). Curran Associates.

[44]. Wu, L., Tordsson, J., Elmroth, E., & Kao, O. (2020). MicroRCA: Root cause localization of performance issues in microservices. In NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium (pp. 1-9). IEEE. https://doi.org/10.1109/NOMS47738.2020.9110353

[45]. Xie, R., Yang, J., Li, J., & Wang, L. (2023). ImpactTracer: Root cause localization in microservices based on fault propagation modeling. In 2023 Design, Automation & Test in Europe Conference & Exhibition (DATE) (pp. 1-6). IEEE. https://doi.org/10.23919/DATE56975.2023.10137236

[46]. Xie, Z., Zhang, S., Geng, Y., Chen, Y., et al. (2023). TraceRank: Abnormal service localization with dis-aggregated end-to-end tracing data in cloud-native systems. Journal of Software: Evolution and Process, 35(10), Article e2413. https://doi.org/10.1002/smr.2413

[47]. Xu, M., Song, C., Wu, H., Gill, S. S., Ye, K., & Xu, C. (2022). esDNN: Deep neural network based multivariate workload prediction in cloud computing environments. ACM Transactions on Internet Technology, 22(3), Article 65. https://doi.org/10.1145/3524114

[48]. Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., Xie, C., Yang, X., Cheng, Q., Li, Z., Chen, J., He, X., Yao, R., Lou, J.-G., Chintalapati, M., Shen, F., & Zhang, D. (2019). Robust log-based anomaly detection on unstable log data. In Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2019) (pp. 807-817). Association for Computing Machinery. https://doi.org/10.1145/3338906.3338931

[49]. Zhang, Z., Ramanathan, M. K., Raj, P., Parwal, A., Sherwood, T., & Chabbi, M. (2022). CRISP: Critical path analysis of large-scale microservice architectures. In 2022 USENIX Annual Technical Conference (USENIX ATC 22) (pp. 655-672). USENIX Association.

[50]. Zharikov, E., Telenyk, S., & Bidyuk, P. (2020). Adaptive workload forecasting in cloud data centers. Journal of Grid Computing, 18(1), 149-168. https://doi.org/10.1007/s10723-020-09507-6