International IT Journal of Research (IITJR), ISSN: 3007-6706
Volume 3, Issue 3, July-September, 2025
Available online at: https://itjournal.org/index.php

Transformer-Based Log Anomaly Detection in Cloud-Native Microservices Using Self-Supervised Pretraining

Dr. Kavita Subramanian

Assistant Professor, Department of Computer Science and Engineering, Indian Institute of Technology Madras, India

ABSTRACT

The rapid adoption of cloud-native architectures and microservices has significantly increased the complexity of modern distributed computing environments. Microservices-based applications generate massive volumes of heterogeneous log data across containers, orchestration platforms, distributed services, and cloud infrastructure. These logs provide essential information for monitoring system behavior, identifying operational failures, and detecting security threats. However, traditional log anomaly detection techniques often struggle to capture complex contextual relationships, temporal dependencies, and evolving behavioral patterns in large-scale distributed environments. This paper proposes a Transformer-Based Log Anomaly Detection Framework using Self-Supervised Pretraining to improve the accuracy, scalability, and efficiency of anomaly identification in cloud-native microservices. The proposed approach leverages transformer architectures to learn contextual representations of log sequences and identify abnormal patterns through attention-based mechanisms. Self-supervised pretraining is employed to learn meaningful log representations from large volumes of unlabeled operational data, reducing dependence on manually annotated datasets. The framework incorporates log parsing, tokenization, semantic embedding, contextual sequence modeling, and anomaly scoring to distinguish normal operational behavior from unexpected events. Furthermore, temporal attention mechanisms are utilized to capture dependencies between log events generated across distributed microservices. The proposed methodology aims to address challenges associated with dynamic service environments, high-dimensional log data, concept drift, and previously unseen anomalies. Performance evaluation can be conducted using standard log anomaly detection datasets and cloud-native application traces, considering metrics such as precision, recall, F1-score, detection latency, and computational overhead. The framework is expected to demonstrate improved generalization capabilities compared with conventional machine learning and deep learning approaches, particularly in scenarios involving limited labeled data. The study highlights the potential of self-supervised transformer models to strengthen intelligent observability, proactive fault diagnosis, and automated incident management in modern cloud-native infrastructures.

Keywords: Transformer Architecture, Log Anomaly Detection, Cloud-Native Microservices, Self-Supervised Learning, Distributed Systems.

INTRODUCTION

The rapid evolution of cloud computing, containerization, and distributed application architectures has transformed the development and deployment of modern software systems. Cloud-native applications increasingly adopt microservices architectures, in which complex applications are divided into smaller, independently deployable services that communicate through application programming interfaces (APIs), messaging systems, and distributed communication protocols. Technologies such as Docker, Kubernetes, and service meshes have further accelerated this transformation by enabling flexible resource allocation, automated orchestration, scalability, and continuous application delivery. Despite these advantages, managing and monitoring cloud-native microservices has become increasingly challenging because of their distributed nature, dynamic execution environments, and complex inter-service dependencies.

Logs represent one of the most important sources of operational information in cloud-native environments. They record application events, service interactions, authentication activities, infrastructure conditions, execution failures, and unexpected system behavior. By analyzing log data, administrators and DevOps engineers can identify performance degradation, detect software defects, investigate security incidents, and maintain service reliability. However, modern microservices continuously generate enormous volumes of heterogeneous log messages, making manual inspection and conventional rule-based monitoring approaches inefficient and difficult to maintain.

Traditional log anomaly detection techniques generally rely on predefined rules, statistical analysis, machine learning algorithms, and supervised classification models. Although these approaches can identify certain abnormal patterns, they frequently encounter limitations when processing high-dimensional log sequences, dynamically changing service behaviors, and previously unseen anomalies. Rule-based approaches require continuous manual updates, while supervised learning techniques depend heavily on accurately labeled training datasets. In practical cloud-native

environments, obtaining sufficient labeled anomalous logs is particularly difficult because system failures and security incidents are relatively rare compared with normal operational events.

Deep learning techniques have introduced significant improvements in automated log analysis. Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Convolutional Neural Networks (CNNs) have been applied to capture sequential and structural relationships within log data. These approaches can identify complex patterns that may not be detected through traditional statistical techniques. Nevertheless, recurrent architectures may experience difficulties in capturing long-range dependencies, while conventional neural networks may require substantial computational resources and domain-specific training procedures.

PROPOSED MODELS AND METHODOLOGIES

The proposed research develops a Transformer-Based Log Anomaly Detection Framework using Self-Supervised Pretraining to identify abnormal operational patterns in cloud-native microservices environments. The methodology combines natural language processing (NLP), transformer-based sequence modeling, self-supervised representation learning, and anomaly scoring techniques. The primary objective is to identify anomalous log sequences accurately while minimizing dependence on manually labeled training data.

The proposed framework is designed to process large-scale log streams generated by distributed microservices, containerized applications, and cloud infrastructure. It learns normal operational behavior from unlabeled log data and identifies deviations that may indicate application failures, security incidents, resource bottlenecks, or communication problems.

2.1 Overall Proposed Framework

The framework consists of seven interconnected processing stages:

1. Log data collection and aggregation.
2. Log preprocessing and normalization.
3. Log parsing and semantic representation.
4. Self-supervised transformer pretraining.
5. Contextual sequence modeling.
6. Anomaly detection and scoring.
7. Performance evaluation and continuous adaptation.

Leveraging Large Language Models and BERT for Log Parsing and Anomaly Detection

2.2 Log Data Collection and Aggregation

The first stage collects operational logs from distributed microservices and cloud-native infrastructure. Logs may originate from application containers, Kubernetes clusters, API gateways, databases, authentication services, and communication middleware.

Potential data sources include:

- Application execution logs.
- Container and orchestration logs.
- API request and response logs.
- Database transaction logs.
- Authentication and authorization logs.
- Network communication logs.
- Infrastructure monitoring events.

Log aggregation tools such as Elasticsearch, Fluentd, Logstash, and OpenTelemetry can be integrated to collect and centralize distributed log information.

Each log record may contain the following attributes:

Attribute	Description
Timestamp	Time of log generation
Service ID	Microservice identifier
Log level	INFO, WARNING, ERROR, DEBUG
Event message	Recorded operational event
Trace ID	Distributed request identifier
Container ID	Container identification
Event template	Normalized log structure

Centralized aggregation provides a unified representation of operational activities across distributed services.

2.3 Log Preprocessing and Normalization

Raw log messages generally contain unstructured text, variable parameters, timestamps, identifiers, and numerical values. These characteristics make direct processing difficult for deep learning models.

The preprocessing stage performs the following operations:

A. Noise Removal: Removes irrelevant characters, redundant spaces, formatting inconsistencies, and unnecessary metadata.

B. Parameter Normalization: Replaces variable values such as IP addresses, timestamps, memory addresses, request identifiers, and numerical parameters with standardized placeholders.

C. Log Parsing: Converts unstructured log messages into structured event templates using parsing techniques such as Drain.

D. Tokenization: Converts structured log messages into tokens that can be processed by transformer architectures.

E. Sequence Construction: Organizes log events into chronological sequences using time windows, service identifiers, and distributed trace information.

2.4 Self-Supervised Transformer Pretraining Model

The central component of the proposed methodology is a transformer-based self-supervised learning model. Unlike conventional supervised approaches, the proposed model learns contextual relationships from unlabeled log sequences.
The transformer encoder uses multi-head self-attention to capture dependencies among log events.
The attention mechanism is mathematically represented as:

$$Attention(Q,K,V)=Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Where:

- Q represents query vectors.
- K represents key vectors.
- V represents value vectors.
- d_k represents the dimensionality of key vectors.

The self-attention mechanism enables the model to identify important relationships between log events, even when they occur at different positions within a sequence.
Self-Supervised Learning Objectives
The proposed framework employs three complementary pretraining objectives.

1. Masked Log Event Prediction

Selected tokens or event components are masked, and the transformer learns to reconstruct the missing information using surrounding context.

$$L_{MLM}=-\sum_{i\in M}\log P(x_i|X_{\backslash M})$$

Where M represents the masked positions and X represents the input log sequence.

RESULTS & ANALYSIS

The proposed Transformer-Based Log Anomaly Detection Framework using Self-Supervised Pretraining is designed to improve the identification of abnormal operational patterns in cloud-native microservices environments. The evaluation focuses on anomaly detection accuracy, contextual learning capability, computational efficiency, detection latency, and generalization across distributed services.

The experimental framework considers log datasets generated from distributed applications, containerized environments, and cloud infrastructure. The proposed transformer model is compared with conventional machine learning approaches, recurrent neural networks, and existing transformer-based log anomaly detection techniques.

Note: The numerical values presented below are illustrative experimental results intended to demonstrate a possible evaluation framework. They are not results from an actually executed experiment.

3.1 Experimental Evaluation Setup

The proposed model can be evaluated using publicly available datasets, including:

- HDFS: Distributed file system logs.
- BGL: Blue Gene/L supercomputer logs.

- Thunderbird: Large-scale system logs.
- OpenStack: Cloud infrastructure logs.

The datasets are preprocessed using log parsing, tokenization, normalization, and sequence construction. The transformer model is pretrained using unlabeled log sequences and subsequently evaluated through anomaly detection tasks.

The following evaluation parameters are considered:

Parameter	Experimental Configuration
Model architecture	Transformer Encoder
Learning approach	Self-Supervised Pretraining
Input representation	Tokenized log sequences
Attention mechanism	Multi-Head Self-Attention
Pretraining objectives	Masked prediction, reconstruction, contextual prediction
Anomaly detection	Combined anomaly scoring
Evaluation metrics	Accuracy, Precision, Recall, F1-score
Baseline models	PCA, Isolation Forest, DeepLog, LogAnomaly, LogBERT

3.2 Anomaly Detection Performance Analysis

The first experiment evaluates the classification performance of the proposed framework against existing anomaly detection approaches.

Table 1. Comparative anomaly detection performance (illustrative values)

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
PCA	88.2	86.5	84.7	85.6
Isolation Forest	90.1	88.4	87.2	87.8
DeepLog	93.4	92.1	91.3	91.7
LogAnomaly	94.2	93.5	92.4	92.9
LogBERT	95.6	95.1	94.3	94.7
Proposed Model	**97.3**	**97.0**	**96.5**	**96.7**

The illustrative results indicate that the proposed framework achieves the highest performance across the four evaluation metrics. Its F1-score of 96.7% suggests a favorable balance between precision and recall.

Compared with LogBERT, the proposed approach demonstrates a hypothetical improvement of 2.0 percentage points in F1-score. This improvement is intended to represent the potential benefit of combining multiple self-supervised learning objectives with cross-service contextual analysis.

3.3 Performance Analysis Across Different Log Datasets

The second experiment examines the generalization capability of the proposed framework across different operational environments.

Table 2. Illustrative F1-score comparison across datasets

Model	HDFS (%)	BGL (%)	Thunderbird (%)	OpenStack (%)
PCA	90.2	82.5	84.3	86.1
Isolation Forest	91.4	85.3	86.7	88.2
DeepLog	94.1	89.7	90.5	91.3
LogAnomaly	94.8	91.2	91.7	92.1
LogBERT	96.0	93.5	94.1	94.4
Proposed Model	**97.2**	**95.8**	**96.1**	**96.5**

The illustrative results suggest that the proposed model maintains comparatively strong performance across heterogeneous log datasets. This indicates the potential of self-supervised pretraining to learn reusable contextual representations from different operational environments.

The improvement across datasets may be particularly useful in cloud-native microservices, where application architectures, logging formats, and service dependencies frequently change.

3.4 Analysis of Detection Latency

Detection latency is an important consideration for real-time anomaly monitoring. Delayed detection may increase service downtime, resource consumption, and operational costs.

Table 3. Illustrative anomaly detection latency

Model	Average Detection Latency (ms)
PCA	42
Isolation Forest	38
DeepLog	55
LogAnomaly	49
LogBERT	67
Proposed Model	**45**

The proposed model demonstrates a hypothetical average detection latency of 45 milliseconds per evaluated sequence. Although its transformer architecture introduces additional computational requirements, efficient inference and sequence batching may help maintain responsive anomaly detection.

Actual latency will depend on hardware configuration, sequence length, batch size, model size, and the number of concurrent microservices.

3.5 Analysis of Self-Supervised Pretraining

An important objective of the proposed framework is to reduce dependence on manually labeled anomaly data.
The following experiment illustrates the potential impact of different proportions of labeled training data.

Table 4. Illustrative impact of labeled data availability

Labeled Training Data	F1-Score (%)
5%	89.4
10%	92.1
20%	94.3
40%	95.8
60%	96.2
100%	96.7

The illustrative results indicate that self-supervised pretraining may enable the model to achieve competitive detection performance even when only a limited proportion of labeled data is available.

COMPARATIVE ANALYSIS IN TABULAR FORM

The comparative analysis evaluates the proposed **Transformer-Based Log Anomaly Detection Framework Using Self-Supervised Pretraining** against existing statistical, machine learning, and deep learning approaches. The comparison focuses on detection accuracy, contextual learning, data requirements, computational efficiency, scalability, and adaptability to dynamic cloud-native microservices environments.

4.1 Comparison of Existing and Proposed Models

Table 1. Comparative analysis of log anomaly detection techniques

Detection Model	Learning Technique	Contextual Learning	Label Requirement	Anomaly Detection Capability
PCA	Statistical learning	Low	Low	Moderate
Isolation Forest	Unsupervised ML	Low	Low	Good
DeepLog	LSTM	Moderate	Moderate	Very Good
LogAnomaly	Deep learning	Moderate	Moderate	Very Good
LogBERT	Transformer	High	Low/Unlabeled pretraining	Excellent
Proposed Model	**Self-Supervised Transformer**	**Very High**	**Reduced labeling requirement**	**Excellent**

4.2 Quantitative Performance Comparison

The following table presents illustrative performance values for comparative evaluation.

Table 2. Quantitative comparison of anomaly detection performance

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
PCA	88.2	86.5	84.7	85.6
Isolation Forest	90.1	88.4	87.2	87.8
DeepLog	93.4	92.1	91.3	91.7
LogAnomaly	94.2	93.5	92.4	92.9
LogBERT	95.6	95.1	94.3	94.7
Proposed Model	**97.3**	**97.0**	**96.5**	**96.7**

Note: *These are illustrative values, not experimentally verified measurements.*

The proposed framework demonstrates the highest hypothetical F1-score of 96.7%, indicating its potential to improve anomaly identification while maintaining a balance between precision and recall.

4.3 Comparison of Computational and Operational Characteristics

Table 3. Computational efficiency and deployment comparison

Parameters	PCA	Isolation Forest	DeepLog	LogBERT	Proposed Model
Training Complexity	Low	Low–Moderate	Moderate	High	High
Inference Complexity	Low	Low	Moderate	High	Moderate–High
Long-Range Dependency	Limited	Limited	Moderate	High	High
Unlabeled Data Utilization	Limited	Yes	Limited	Yes	Yes
Cross-Service Analysis	Limited	Limited	Limited	Possible	Integrated
Dynamic Environment Adaptation	Limited	Moderate	Moderate	Moderate	Designed for adaptation
Scalability	High	High	Moderate	High with optimization	High with optimization
Contextual Representation	Low	Low	Moderate	High	High

4.4 Dataset-Wise Comparative Analysis

Table 4. Comparison of F1-score across different log datasets

Model	HDFS (%)	BGL (%)	Thunderbird (%)	OpenStack (%)
PCA	90.2	82.5	84.3	86.1
Isolation Forest	91.4	85.3	86.7	88.2
DeepLog	94.1	89.7	90.5	91.3
LogAnomaly	94.8	91.2	91.7	92.1
LogBERT	96.0	93.5	94.1	94.4
Proposed Model	**97.2**	**95.8**	**96.1**	**96.5**

Note: *Dataset-wise values are illustrative and require validation through actual experiments.*

The comparison suggests that the proposed framework may offer improved generalization across heterogeneous log environments. Its multi-objective pretraining strategy is intended to capture both semantic and temporal relationships in distributed operational logs.

4.5 Comparison of Self-Supervised Learning Capabilities

Table 5. Analysis of pretraining strategies

Learning Strategy	Learning Objective	Contextual Understanding	Expected Benefit
Masked Log Prediction	Predict missing log tokens	High	Learns contextual event representations
Sequence Reconstruction	Reconstruct original sequences	High	Identifies deviations from normal patterns
Contextual Event Prediction	Predict subsequent events	High	Captures temporal dependencies
Cross-Service Contextual Learning	Learn inter-service relationships	Very High	Identifies cascading failures
Combined Proposed Strategy	**Multi-objective self-supervised learning**	**Very High**	**Comprehensive anomaly representation**

LIMITATIONS & DRAWBACKS

Although the proposed Transformer-Based Log Anomaly Detection Framework Using Self-Supervised Pretraining offers significant potential for improving anomaly identification in cloud-native microservices, several technical, operational, and computational limitations must be considered. The integration of transformer architectures, self-supervised learning, and distributed log analysis introduces challenges related to computational complexity, data quality, scalability, model interpretability, and deployment.

5.1 High Computational Requirements

Transformer-based architectures generally require substantial computational resources for training and inference. Processing large-scale log sequences from multiple microservices can increase GPU memory consumption, processing time, and infrastructure costs. Smaller organizations may face difficulties deploying resource-intensive models.

5.2 Complexity of Log Data Preprocessing

Cloud-native environments generate heterogeneous and unstructured log messages. Differences in logging formats, timestamps, service identifiers, and event structures can affect preprocessing quality. Incorrect log parsing or normalization may result in information loss and reduced anomaly detection accuracy.

5.3 Dependence on Quality of Pretraining Data

Although self-supervised learning reduces the need for manually labeled datasets, its effectiveness depends on the quality and diversity of available log data. If the pretraining dataset contains incomplete, biased, or noisy information, the model may learn inaccurate representations of normal system behavior.

5.4 False Positive and False Negative Detection

The proposed framework may incorrectly classify legitimate operational variations as anomalies or fail to identify subtle abnormal patterns. False positives can generate unnecessary alerts, while false negatives may allow critical failures or security incidents to remain undetected.

5.5 Difficulty in Detecting Concept Drift

Microservices environments continuously evolve because of software updates, configuration changes, workload variations, and infrastructure modifications. Consequently, normal log patterns may change over time. A model trained on historical data may become less effective when previously unseen operational patterns emerge.

CONCLUSION

The study, **"Transformer-Based Log Anomaly Detection in Cloud-Native Microservices Using Self-Supervised Pretraining,"** presents an intelligent framework for identifying abnormal operational patterns in complex and distributed cloud-native environments. The increasing adoption of microservices architectures, containerization, and cloud computing has significantly increased the volume, diversity, and complexity of system-generated logs. Traditional anomaly detection approaches often face challenges in processing large-scale log sequences, capturing long-range dependencies, identifying previously unseen anomalies, and adapting to continuously changing application environments.

To address these challenges, the proposed framework integrates transformer-based contextual learning with self-supervised pretraining techniques. The methodology incorporates log collection, preprocessing, normalization,

tokenization, contextual representation learning, and anomaly scoring. The use of masked log prediction, sequence reconstruction, and contextual event prediction enables the model to learn meaningful representations from large volumes of unlabeled operational data, thereby reducing dependence on manually annotated datasets.

The proposed transformer architecture utilizes multi-head self-attention mechanisms to identify complex relationships among log events and capture temporal dependencies across distributed microservices. Furthermore, cross-service dependency analysis provides opportunities to identify cascading failures, abnormal communication patterns, and unexpected service interactions. The combined anomaly scoring mechanism is designed to improve the identification of deviations from normal operational behavior.

The comparative evaluation framework considers established approaches, including PCA, Isolation Forest, DeepLog, LogAnomaly, and LogBERT. The illustrative results indicate the potential of the proposed methodology to achieve improved anomaly detection performance, particularly in terms of precision, recall, and F1-score. However, these numerical findings require validation through actual experimental implementation using standardized datasets and reproducible testing environments.

Despite its potential advantages, the proposed framework involves challenges related to computational complexity, model interpretability, data quality, concept drift, and real-time processing requirements. Future research should therefore investigate lightweight transformer architectures, adaptive self-supervised learning, explainable AI, federated learning, and graph-based service dependency modeling to improve practical deployment.

In conclusion, transformer-based self-supervised pretraining represents a promising approach to intelligent log anomaly detection in cloud-native microservices. By combining contextual sequence understanding, reduced labeling requirements, and distributed service analysis, the proposed framework provides a foundation for developing scalable and adaptive monitoring systems. Its successful implementation could strengthen cloud infrastructure reliability, improve fault diagnosis, reduce operational disruptions, and support proactive incident management. Further research and real-world validation will be essential to establish its effectiveness across diverse cloud platforms and large-scale distributed application environments.

REFERENCES

1. Ali, S., Boufaied, C., Bianculli, D., Branco, P., & Briand, L. (2025). A comprehensive study of machine learning techniques for log-based anomaly detection. *Empirical Software Engineering*, *30*, Article 129. https://doi.org/10.1007/s10664-025-10669-3
2. Bertero, C., Roy, M., Sauvanaund, C., & Trédan, G. (2017). Experience report: Log mining using natural language processing and application to anomaly detection. In *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)* (pp. 351–360). IEEE.
3. Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data* (pp. 93–104). ACM. https://doi.org/10.1145/342009.335388
4. 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
5. Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, *41*(3), Article 15. https://doi.org/10.1145/1541880.1541882
6. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423
7. 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). ACM. https://doi.org/10.1145/3133956.3134015
8. Guo, H., Yuan, S., & Wu, X. (2021). LogBERT: Log anomaly detection via BERT. In *2021 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8). IEEE. https://doi.org/10.1109/IJCNN52387.2021.9534113
9. He, P., Zhu, J., Zheng, Z., & Lyu, M. R. (2017). Drain: An online log parsing approach with fixed depth tree. In *2017 IEEE International Conference on Web Services (ICWS)* (pp. 33–40). IEEE. https://doi.org/10.1109/ICWS.2017.13
10. 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
11. Jamshidi, P., Pahl, C., Mendonça, N. C., Lewis, J., & Tilkov, S. (2018). Microservices: The journey so far and challenges ahead. *IEEE Software*, *35*(3), 24–35. https://doi.org/10.1109/MS.2018.2141039

12. 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/ASE51524.2021.9678798
13. Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining* (pp. 413–422). IEEE. https://doi.org/10.1109/ICDM.2008.17
14. 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* (pp. 4739–4745). International Joint Conferences on Artificial Intelligence. https://doi.org/10.24963/ijcai.2019/658
15. Newman, S. (2021). *Building microservices: Designing fine-grained systems* (2nd ed.). O'Reilly Media.
16. Oliner, A., & Stearley, J. (2007). What supercomputers say: A study of five system logs. In *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*. ACM. https://doi.org/10.1109/SC.2007.66
17. Pahl, C. (2015). Containerization and the PaaS cloud. *IEEE Cloud Computing, 2*(3), 24–31. https://doi.org/10.1109/MCC.2015.51
18. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems, 30*. Curran Associates, Inc.
19. Xu, W., Huang, L., Fox, A., Patterson, D., & Jordan, M. I. (2009). Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles* (pp. 117–132). ACM. https://doi.org/10.1145/1629575.1629587
20. Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., Xie, C., Yang, X., Cheng, Q., Li, Z., Chen, J., Chintalapati, M., Zhang, D., Rajmohan, S., & Lin, 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* (pp. 807–817). ACM. https://doi.org/10.1145/3338906.3338931