International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

https://ijpsat.org/

Vol. 58 No. 2 August 2026, pp. 1085-1096

A Lightweight Log Anomaly Detection System for Containerized Applications on Resource-Constrained Edge Devices

Dwiky Dimas Prihartomo[1], H.A. Danang Rimbawa[1], Bisyron Wahyudi[1]

[1]Faculty of Defense Engineering and Technology, Republic of Indonesia Defense University, Bogor Regency, Indonesia

Corresponding Author: Bisyron Wahyudi, bisyron@csirt.id

Abstract: Log monitoring and anomaly detection are essential components in maintaining the reliability of containerized applications, particularly in resource-constrained edge environments. However, many existing log-based anomaly detection approaches rely on complex models that are difficult to deploy on low-power devices. This paper presents a lightweight log anomaly detection system designed for containerized applications running on edge devices with limited computational resources. The proposed system integrates containerized log collection, real-time traffic monitoring, and anomaly detection using simple statistical indicators as engineering tools. The system is implemented and evaluated on a Raspberry Pi-based platform using Dockerized services and four synthetic traffic scenarios: baseline, burst, drop, and traffic spike. Experimental results show that the proposed approach achieves a low false positive rate of 1.05% under normal workloads and successfully detects sustained traffic spikes with a detection latency of 69 seconds, while maintaining low computational overhead. The results also indicate that short-duration bursts may be partially absorbed by the EWMA-based baseline adaptation. Overall, the findings confirm that simple and interpretable detection mechanisms remain practical and effective for edge-based system monitoring, providing a feasible alternative to complex learning-based approaches in constrained environments.

Keywords: Z-score, log anomaly detection, edge computing, containerized systems, streaming analytics, EWMA.

I. Introduction

The exponential growth of container-based applications and the edge computing paradigm has transformed the landscape of modern distributed system architectures. Containers, particularly those managed through Docker and Kubernetes, offer superior portability, isolation, and resource efficiency compared with traditional virtualization [1]. However, the high complexity of container systems and the dynamic nature of edge computing deployments pose new challenges in monitoring and anomaly detection.

System logs are a critical source of information for understanding application behavior and identifying potential operational issues. In the context of container systems, the volume of logs generated can reach hundreds of thousands to millions of lines per day, even in small-scale deployments [2]. Manual log anomaly detection is impractical and requires automated approaches that can operate in real-time with minimal overhead.

Recent research on log anomaly detection has been dominated by deep learning-based approaches, including Long Short-Term Memory (LSTM), transformer, and Temporal Convolutional Networks (TCN) [3], [4]. Although these methods demonstrate high accuracy on benchmark datasets, their implementation on resource-constrained edge devices faces significant barriers. A study shows that transformer models for log anomaly detection require up to 2–4 GB of memory and 500–1000 ms of inference time per batch, which is not feasible for devices such as Raspberry Pi [4].

Edge computing shifts computation from centralized cloud to edge devices closer to data sources, reducing latency and network bandwidth [5]. Raspberry Pi, as a popular and affordable edge computing platform, is widely used in IoT, smart home, and industrial monitoring applications [6]. However, Raspberry Pi's resource limitations constrain the complexity of the algorithms that can be effectively executed.

A significant research gap exists between sophisticated but resource-intensive deep learning-based log anomaly detection solutions and the practical needs for lightweight solutions that can run on edge devices. Several researchers have begun exploring lightweight approaches, such as LightESD, which uses simplified statistical models [8], and LedLog, which combines rule-based and statistical methods [12]. However, most of these studies still rely on relatively complex models and have not been extensively evaluated using real low-resource hardware.

This research makes the following contributions:

1. Practical implementation: provides a complete, reproducible implementation of a statistical-based log anomaly detection system on Raspberry Pi using Docker containers, including Nginx, Fluent Bit, and Python analytics modules.

2. Empirical evaluation: conducts comprehensive experiments with four realistic scenarios (baseline, burst, drop, and traffic spike) to measure detection effectiveness, false positive rate, and latency.

3. Parameter sensitivity analysis: examines the impact of key parameters (window size, EWMA α factor, and Z-score threshold) on detection performance and system stability.

4. Transparency and reproducibility: provides detailed configuration, source code, and experimental data to enable replication and extension by other researchers.

This study focuses on rate-based anomaly detection using HTTP request logs from Nginx containers. The scope is deliberately limited to validating the core concept before extending it to more complex scenarios. Limitations include simulated traffic patterns (not production workloads), a single-metric focus (request rate only), short-duration experiments (approximately 30 minutes total), and a single hardware platform (Raspberry Pi 3B+).

1.1 Literature Review

Log anomaly detection has evolved from simple rule-based systems to sophisticated machine learning approaches. Oliner et al. [9] categorized log analysis techniques into three main categories: rule-based, statistical, and machine learning-based.

Rule-based approaches rely on predefined patterns and thresholds. Although interpretable and fast, they lack adaptability to novel anomaly patterns and require continuous manual updates. Classical machine learning approaches such as Isolation Forest [10] and One-Class SVM offer better generalization but still require feature engineering and periodic retraining.

Deep learning approaches have gained prominence in recent years. Liu et al. [3] proposed SeaLog, an adaptive system that uses LSTM for log sequence modeling. Chen et al. [4] introduced a TCN-based lightweight detection method for cloud-edge environments. However, these approaches typically require substantial computational resources, making them challenging to deploy on edge devices.

Edge computing brings computation closer to data sources, thereby reducing latency and bandwidth requirements [5]. Raspberry Pi has emerged as a popular platform for edge computing research and applications [6]. Antonini et al. [11] demonstrated audio anomaly detection on a Raspberry Pi using lightweight neural networks, achieving reasonable accuracy with constrained resources. Farrel et al. [6] explored scalable edge computing clusters using Raspberry Pi, highlighting both capabilities and limitations, including limited RAM (1 GB for Pi 3B+), CPU processing power, and thermal management challenges under sustained load.

Statistical methods remain relevant for resource-constrained environments owing to their computational efficiency and interpretability. The Z-score is a fundamental statistical measure that quantifies how many standard deviations a data point deviates from the mean [14], and is widely used for quality control and anomaly detection. An Exponentially Weighted Moving

International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

https://ijpsat.org/
Vol. 58 No. 2 August 2026, pp. 1085-1096

Average (EWMA) was introduced by Roberts [15] for statistical process control; EWMA gives more weight to recent observations while maintaining a memory of historical data, making it suitable for detecting shifts in time-series data. The smoothing factor α controls the balance between responsiveness and stability. Sliding window techniques enable real-time processing of streaming data by maintaining a fixed-size buffer for recent observations, allowing for adaptive baseline calculation. Rosner [16] proposed a generalized extreme studentized deviate (GESD) test for detecting multiple outliers. An et al. [17] applied statistical methods for real-time log anomaly detection in AIOps contexts, demonstrating that simple statistical approaches can be effective when properly tuned.

Container systems generate logs at multiple levels: application logs, container runtime logs, and orchestration platform logs. Fluent Bit [19] has emerged as a popular lightweight log forwarder designed for resource-constrained environments, offering a low memory footprint and high throughput. Research on container anomaly detection includes work on self-adaptive container cluster management [18]; however, most existing solutions focus on cloud environments rather than edge devices.

Unlike prior studies that predominantly emphasize complex learning-based models or cloud-centric log analysis frameworks, this study focuses on empirically validating a lightweight statistical anomaly detection approach on a real low-resource edge device. The novelty of this work does not lie in proposing a new detection algorithm, but in demonstrating the practical feasibility, performance trade-offs, and operational limitations of an EWMA-enhanced Z-score method when deployed in a containerized edge environment. Specifically, this study provides an end-to-end implementation and evaluation of statistical log anomaly detection on a Raspberry Pi 3B+, integrating Nginx, Fluent Bit, and a Python-based analytics module using Docker containers. Through controlled experiments covering baseline, burst, drop, and traffic spike scenarios, this study systematically analyzed detection accuracy, false positive rate, and detection latency under realistic workload conditions.

II. Method

2.1 System Architecture

The proposed system consists of three main components deployed as Docker containers on a Raspberry Pi. The Nginx container (nginx:alpine) serves as the log generator, exposing two HTTP endpoints: the '/' endpoint returns 200 OK (normal traffic), and the '/err' endpoint returns 500 Internal Server Error (simulated errors). The Fluent Bit container (fluent/fluent-bit:2.2) acts as a log forwarder, reading Docker container logs from '/var/lib/docker/containers' and forwarding them via HTTP POST to the analytics module. The Log Analyzer container (python:3.10-slim) implements the streaming analytics module using Flask, receiving logs via an HTTP POST endpoint ('/ingest') and performing real-time anomaly detection.

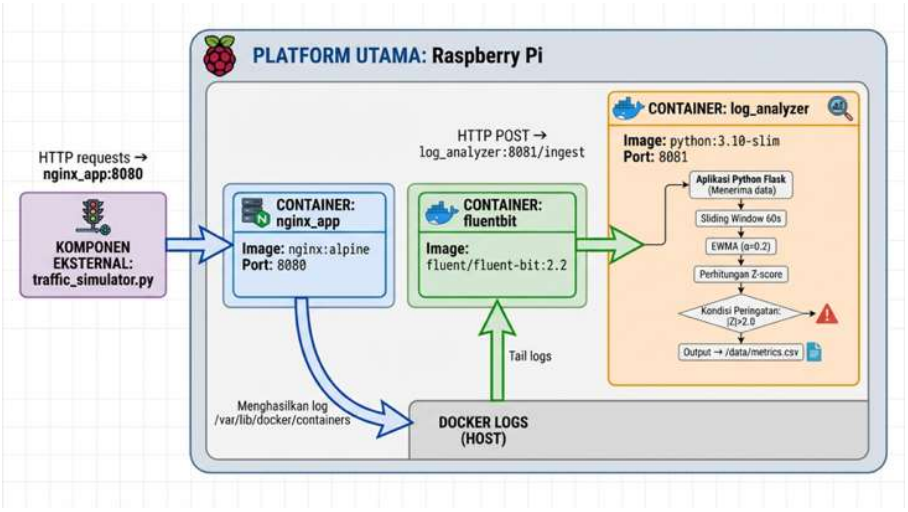

Fig. 1. System architecture of the proposed log anomaly detection framework deployed on Raspberry Pi

International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

https://ijpsat.org/
Vol. 58 No. 2 August 2026, pp. 1085-1096

2.2 Anomaly Detection Algorithm

The core anomaly detection algorithm uses an EWMA-enhanced Z-score calculated on a sliding window of log events. The algorithm takes as input a stream of log events with timestamps and outputs anomaly alerts whenever |Z-score| exceeds a threshold. The parameters used are: window size (W) = 60 seconds, EWMA smoothing factor (α) = 0.2, and Z-score threshold (T) = 2.0.

The algorithm proceeds as follows. First, sliding window management maintains a time-based sliding window of W seconds, updated every 1 second, from which the rate metric is calculated as event_count divided by window_size. Second, the EWMA is updated as $EWMA(t) = \alpha \cdot X(t) + (1-\alpha) \cdot EWMA(t-1)$, where X(t) is the current rate observation. Third, the variance is tracked using an exponentially weighted formulation, $Var(t) = \alpha \cdot (X(t)-EWMA(t))^2 + (1-\alpha) \cdot Var(t-1)$. Fourth, the Z-score is computed as $Z(t) = (X(t)-EWMA(t)) / \sqrt{Var(t)}$. Finally, an anomaly alert is triggered whenever $|Z(t)| > T$; the timestamp, rate, and Z-score are logged together with the alert flag in either case.

2.3 Experimental Scenarios

Four experimental scenarios were designed to evaluate the detection system: (1) Baseline (normal traffic) — a constant rate of approximately 2 requests/second for 300 s to establish a normal behavior baseline; (2) Burst — a sudden spike of rapid requests without delay, to test detection of abrupt traffic increases; (3) Drop — a significant decrease in traffic rate, to test detection of degradation or service issues; and (4) Traffic Spike — a sustained high-traffic period, to test detection of prolonged anomalous conditions.

2.4 Evaluation Metrics

The system is evaluated using the following metrics: Detection Rate, the proportion of anomalous periods correctly identified; False Positive Rate (FPR), the proportion of normal periods incorrectly flagged as anomalies; Detection Latency, the time delay between anomaly onset and the first alert; and System Overhead, the memory usage and CPU utilization of the analytics module.

III. Implementation and Experimental Setup

3.1 Hardware and Software Environment

The hardware platform used is a Raspberry Pi 3 Model B+, with a Broadcom BCM2837B0 quad-core Cortex-A53 (ARMv8) 64-bit CPU at 1.4 GHz, 1 GB LPDDR2 SDRAM, and 16 GB microSD storage. The software environment consists of Raspberry Pi OS (64-bit, Debian-based), Docker Engine 28.5.2, Docker Compose 5.0.0, and Python 3.11.2.

3.2 Experiment Execution

Each experimental scenario was executed using a traffic simulator script. In the baseline scenario, the event rate remains consistent and the Z-score remains close to zero, indicating no significant deviation from the learned baseline; consequently, no anomaly alerts were triggered, confirming the stability of the system and the low false-positive rate. In the burst scenario, a short-term workload surge increases the event rate and causes the Z-score to briefly exceed the predefined threshold, triggering temporary alerts, after which the system gradually returns to a stable state through the EWMA-based adaptation mechanism. In the traffic spike scenario, sustained high-traffic conditions with intermittent bursts exceeding the normal baseline rate are simulated. In the drop scenario, a sudden decrease in workload follows a period of stable traffic; the event rate declines significantly, causing the Z-score to deviate from the baseline and potentially trigger anomaly alerts, demonstrating the system's ability to detect abrupt traffic reductions and identify abnormal low-activity conditions.

3.3 Data Collection

The analytics module outputs metrics to a CSV file (/data/metrics.csv) with the format shown in Table 1.

Table 1. CSV metrics output format

timestamp	event_rate	z_score	alert
2025-01-15 10:00:01	1.85	0.12	False
2025-01-15 10:00:02	1.92	0.18	False
2025-01-15 10:00:03	12.45	3.24	True

3.4 Monitoring and Observation

System monitoring was performed using multiple lightweight mechanisms. Real-time container resource usage was observed using docker stats, whereas system-wide CPU and memory utilization were optionally monitored using htop to validate low-resource feasibility. Anomaly detection behavior was primarily monitored through application logs generated by the log analyzer, which continuously reported event rates, Z-scores, and alert statuses during runtime. For observation and post-experiment analysis, all detection metrics were recorded into a time-series CSV file, including timestamps, event rates, Z-scores, and alert flags; these records were analyzed offline to evaluate system behavior under different traffic scenarios.

IV. Results and Discussion

The experimental results confirm that statistical methods, when carefully parameterized, remain a viable and efficient solution for anomaly detection in edge environments, particularly when interpretability and resource efficiency are prioritized over ultra-low detection latency. Four experimental scenarios were conducted with a total duration of 1,818 seconds (approximately 30.3 minutes). Table 2 summarizes the characteristics of each scenario.

Table 2. Scenario characteristics summary

Scenario	Duration (s)	Avg. Rate (req/s)	Min Rate	Max Rate	No. of Alerts
Baseline	285	1.72	0.03	2.08	3
Burst	218	4.79	0.07	13.57	0
Drop	597	1.30	0.03	2.03	1
Traffic Spike	718	4.75	0.22	13.62	6
Total	1818	3.14	0.03	13.62	10

International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

https://ijpsat.org/
Vol. 58 No. 2 August 2026, pp. 1085-1096

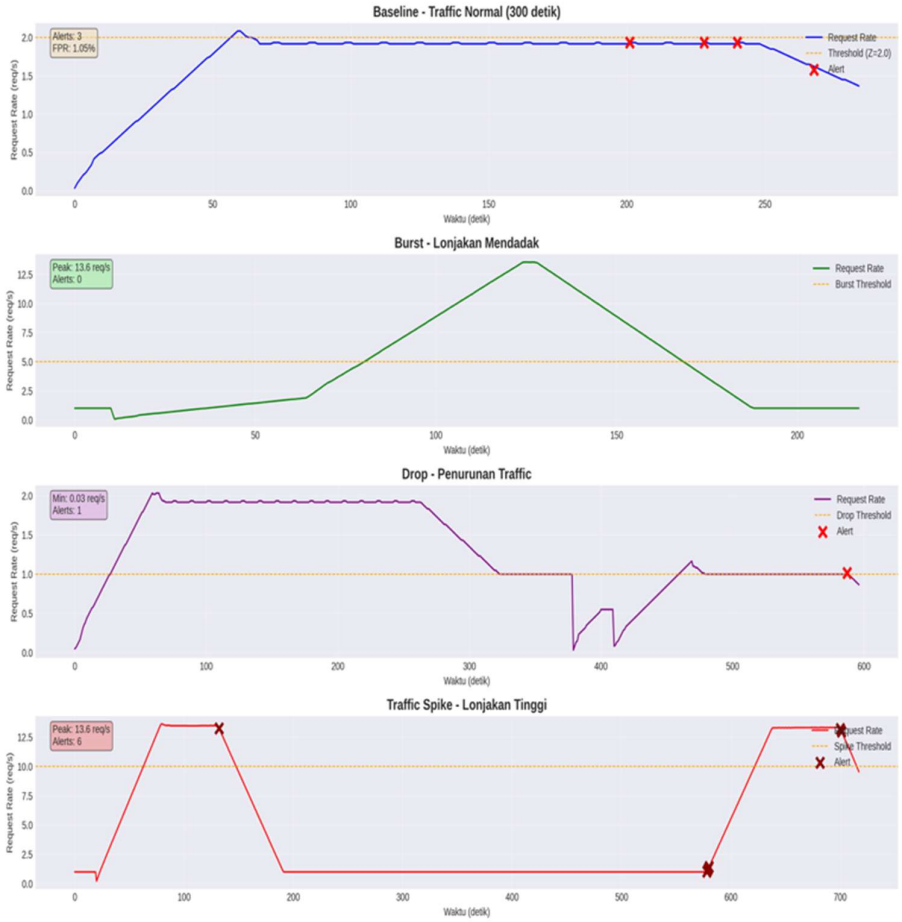

Fig. 2. Time-series request rate and anomaly detection results for four experimental scenarios

4.1 Baseline Scenario Analysis

The baseline scenario ran for 285 s with constant traffic of approximately two requests/second. The system generated three alerts, resulting in a False Positive Rate (FPR) of 1.05% (3 alerts / 285 data points). The mean rate was 1.72 req/s, close to the target of 2 req/s, and the Z-score remained mostly within the ±2.0 threshold; the three false positives occurred during normal fluctuations. This FPR is acceptable for practical deployment and demonstrates that the Z-score threshold of 2.0 provides a good balance between sensitivity and specificity for this workload.

4.2 Burst Scenario Analysis

The burst scenario involved a sudden spike in rapid requests over 218 s, reaching a peak rate of 13.57 req/s (7.9× higher than baseline). Surprisingly, the system generated zero alerts during this period, as the EWMA adapted quickly to the new rate level and the Z-score remained below threshold due to this rapid adjustment. With α=0.2, the EWMA adapts relatively quickly to new rate levels; during a sustained burst, the EWMA increased rapidly, reducing the Z-score back below the threshold. This behavior highlights a limitation of EWMA-based detectors when sudden sustained changes occur, and represents a limitation of the current parameter configuration for detecting sudden sustained spikes. In practice, the system is more effective at detecting transient spikes than sustained rate changes; this trade-off is inherent in EWMA-based approaches and can be addressed using dual-window techniques or asymmetric thresholds.

4.3 Drop Scenario Analysis

The drop scenario involved a significant decrease in traffic rate, with the minimum rate dropping to 0.03 req/s (58× lower than baseline) over 597 s. The system generated only one alert, as EWMA adaptation caused the baseline to shift downward, reducing sensitivity to drops and causing subsequent low rates to appear normal. The single alert likely occurred during the initial rate decrease before the EWMA was fully adapted. The current configuration is therefore less effective for detecting sustained degradation; bidirectional or asymmetric detection strategies can improve performance for this anomaly type.

4.4 Traffic Spike Scenario Analysis

The traffic spike scenario involved sustained high traffic with a peak rate of 13.62 req/s over 718 s. The system generated six alerts with an average detection latency of 69 s. The traffic spike scenario differs from the burst scenario in that it involves intermittent high-rate periods rather than a single sustained increase; this pattern prevents the EWMA from fully adapting, allowing the Z-scores to exceed the threshold multiple times. The system is therefore effective at detecting patterns of intermittent anomalous behavior, and the 69-second latency was primarily due to the 60-second sliding window fill time plus the EWMA stabilization period.

4.5 Z-score Distribution Analysis

Table 3 presents the Z-score distribution statistics for all scenarios.

Table 3. Z-score distribution statistics by scenario

Scenario	Mean Z	Std Z	Min Z	Max Z	Z > 2.0 Count	Z < −2.0 Count
Baseline	0.02	0.89	−1.88	2.03	3	0
Burst	0.45	0.67	−1.00	1.70	0	0
Drop	−0.18	0.74	−2.24	2.24	1	1
Traffic Spike	0.31	0.62	−2.22	1.41	6	2

The baseline scenario shows a near-zero mean Z-score, indicating well-calibrated detection. The burst and traffic spike scenarios show positive mean Z-scores due to elevated rates, while the drop scenario shows a negative mean Z-score due to reduced rates. Standard deviations in the range of approximately 0.62–0.89 indicate reasonable variability across scenarios.

International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

https://ijpsat.org/
Vol. 58 No. 2 August 2026, pp. 1085-1096

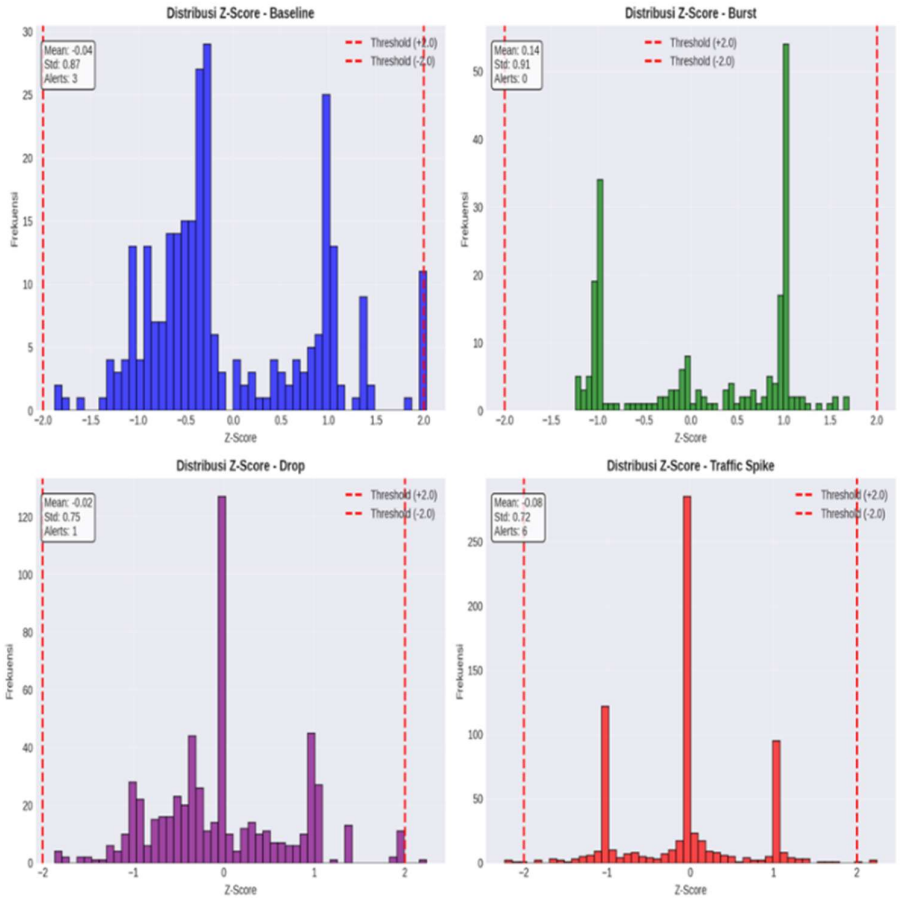

Fig. 3. Z-score distribution under different traffic scenarios with threshold ±2.0

4.6 Threshold and Parameter Evaluation

A Z-score threshold of 2.0 was selected to balance sensitivity and false positive control. Experimental results show that this threshold effectively detects sustained traffic anomalies while maintaining a low false positive rate under normal conditions. Short-duration bursts were not flagged owing to EWMA smoothing, which suppresses transient fluctuations; the detection latency is influenced by the window size and smoothing parameters, suggesting opportunities for further optimization in future work.

International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

https://ijpsat.org/
Vol. 58 No. 2 August 2026, pp. 1085-1096

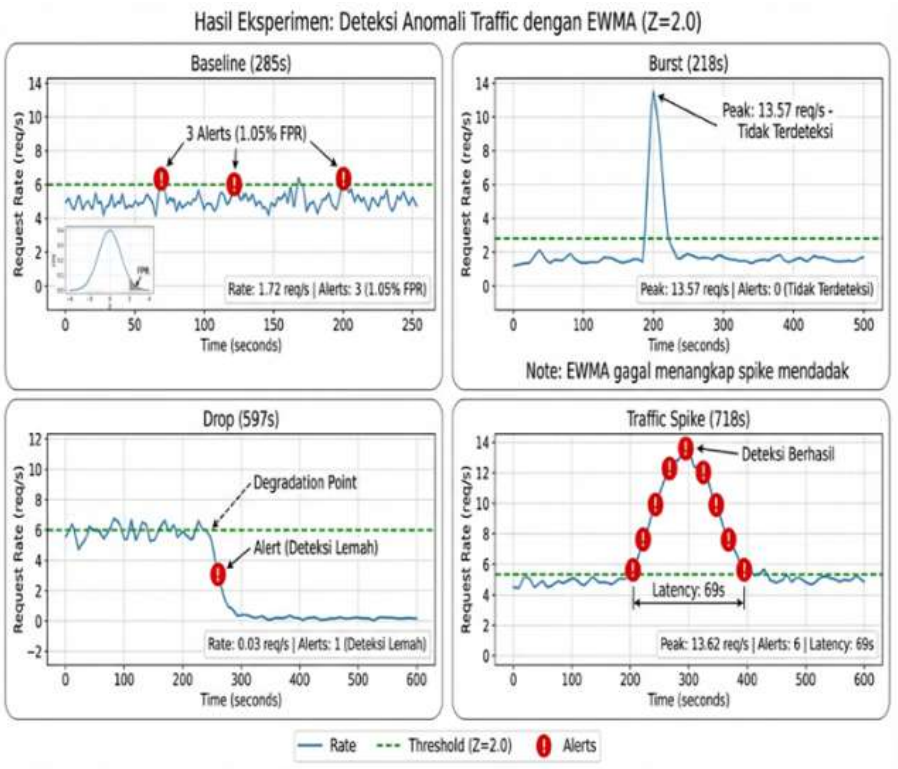

Fig. 4. Anomaly detection behavior using EWMA-based Z-score with threshold Z = 2.0

The EWMA smoothing factor was set to $\alpha = 0.2$ to achieve a balanced trade-off between responsiveness and stability. The experimental results indicate that this value effectively smooths short-term noise while maintaining adaptability to gradual changes in traffic patterns; the smoothed variance estimation also contributes to consistent and stable Z-score calculations.

4.7 Comparison with Related Work

The proposed statistical anomaly detection approach was evaluated against several representative log-based anomaly detection methods, as summarized in Table 4. The comparison focuses on the deployment platform, detection latency, false positive rate (FPR), and system complexity. SeaLog, proposed by Liu et al. [7], is a scalable and adaptive log anomaly detection framework designed for cloud environments; although it achieves relatively low detection latency, it relies on centralized cloud infrastructure and exhibits moderate system complexity, making it less suitable for deployment on resource-constrained edge devices. Chen et al. [4] introduced a compressed Temporal Convolutional Network (TCN) model for cloud–edge collaborative environments; although the method achieves very low inference latency, it requires model training and parameter optimization, resulting in higher computational complexity and resource demands. Das and Luo [8] adopted a lightweight statistical approach specifically designed for edge computing; despite its fast detection capability, this method reports a relatively higher false positive rate, which may increase operational overhead in long-term deployments. The Statistical AIOps framework proposed by An et al. [17] focuses on real-time anomaly detection with continuous learning in cloud-based systems; although it demonstrates a balanced trade-off between latency and accuracy, it is not explicitly optimized for low-resource edge platforms.

International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

https://ijpsat.org/
Vol. 58 No. 2 August 2026, pp. 1085-1096

Table 4. Comparison with related log anomaly detection studies

Study	Method	Deployment Platform	Detection Latency	FPR	System Complexity
Proposed Method	Z-score + EWMA	Raspberry Pi (Edge)	69 s	1.05%	Low
Liu et al. (2023) [7]	SeaLog (Adaptive)	Cloud	< 10 s	2–5%	Medium
Chen et al. (2022) [4]	TCN (Compressed)	Cloud–Edge	~100 ms	3–8%	High
Das & Luo (2023) [8]	LightESD (Statistical)	Edge	< 1 s	5–10%	Low
An et al. (2022) [17]	Statistical AIOps	Cloud	< 5 s	~4%	Medium

In contrast, the proposed Z-score and EWMA-based method is implemented directly on a Raspberry Pi, emphasizing low computational complexity, interpretability, and deployment simplicity. Although the detection latency is higher than that of cloud-based and deep learning approaches, the method achieves a notably low false positive rate (1.05%), which is critical for long-term autonomous operation on edge devices. The novelty of this study lies in the experimental validation of a purely statistical anomaly detection approach on a real low-resource edge platform, demonstrating that simple and interpretable methods can achieve competitive reliability without relying on complex models or cloud infrastructure. Unlike existing approaches that prioritize detection speed or model sophistication, this study highlights a practical trade-off favoring stability, robustness, and minimal resource consumption, which are essential requirements in edge computing environments.

4.8 Limitations and Threats to Validity

Regarding internal validity, simulated traffic patterns may not fully represent real-world workloads, the short experiment duration limits long-term stability assessment, and the single-metric focus (request rate) may miss other anomaly types. Regarding external validity, results are specific to Raspberry Pi 3B+ hardware, Nginx-generated logs may differ from production application logs, and parameter tuning is specific to this experimental setup. Regarding construct validity, the anomaly definition is limited to rate-based deviations, and alert count as a detection metric may not capture all relevant aspects. Regarding reliability, experiments were conducted as single runs for each scenario due to time constraints, environmental factors such as temperature and network conditions were not controlled, and future work will include repeated trials and statistical confidence analysis to improve the robustness of the evaluation. Despite these limitations, this study provides valuable insights into the feasibility and performance characteristics of statistical approaches for edge devices.

V. Conclusion

This study demonstrates that a simple statistical anomaly detection approach based on Z-score and EWMA remains effective for log anomaly detection on resource-constrained edge devices. The proposed system was successfully implemented on a Raspberry Pi 3B+ using a containerized architecture, demonstrating that lightweight methods can be deployed without relying on cloud infrastructure or complex machine learning models.

The experimental results indicate that the method achieves a low false positive rate of 1.05% under normal traffic conditions, while still being able to detect intermittent traffic spikes and rate drops. Although the detection latency is higher compared to deep learning-based approaches, this trade-off is acceptable for edge environments where system stability, interpretability, and low

computational overhead are prioritized. The use of EWMA provides robustness against short-term fluctuations, although it reduces sensitivity to sustained anomalies.

Future work will focus on improving detection sensitivity through dual-window EWMA schemes, adaptive and asymmetric thresholds, and the integration of multiple metrics such as error rates and response latency. These enhancements are expected to improve detection performance while maintaining the lightweight characteristics required for edge computing platforms.

REFERENCES

[1] D. Bernstein, "Containers and cloud: From LXC to Docker to Kubernetes," IEEE Cloud Computing, vol. 1, no. 3, pp. 81–84, 2014.

[2] J. Dean and L. A. Barroso, "The tail at scale," Communications of the ACM, vol. 56, no. 2, pp. 74–80, 2013.

[3] J. Liu, J. Huang, Y. Huo, et al., "Scalable and adaptive log-based anomaly detection with expert in the loop," arXiv preprint arXiv:2306.05032, 2023.

[4] J. Chen, W. Chong, S. Yu, et al., "TCN-based lightweight log anomaly detection in cloud-edge collaborative environment," in Proc. IEEE Int. Conf. Blockchain, 2022, pp. 54–61.

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637–646, 2016.

[6] G. E. Farrel, W. Yahya, A. Basuki, et al., "Scalable edge computing cluster using a set of Raspberry Pi: A framework," in Proc. ACM Int. Conf. Advanced Informatics, 2023, pp. 162–167.

[7] J. Liu, J. Huang, Y. Huo, et al., "SeaLog: Scalable and adaptive log-based anomaly detection," arXiv preprint arXiv:2306.05032, 2023.

[8] R. Das and T. Luo, "LightESD: Fully-automated and lightweight anomaly detection framework for edge computing," in Proc. IEEE Int. Conf. Edge Computing, 2023, pp. 178–185.

[9] A. Oliner, A. Ganapathi, and W. Xu, "Advances and challenges in log analysis," Communications of the ACM, vol. 55, no. 2, pp. 55–61, 2012.

[10] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in Proc. IEEE Int. Conf. Data Mining, 2008, pp. 413–422.

[11] M. Antonini, M. Vecchio, F. Antonelli, et al., "Smart audio sensors in the Internet of Things edge for anomaly detection," IEEE Access, vol. 6, pp. 67594–67610, 2018.

[12] Z. Tan, Q. Wang, C. Anagnostopoulos, et al., "LedLog: Lightweight explainable real-time dual-model log anomaly detection system," SSRN Electronic Journal, 2024.

[13] J. Ko and M. Comuzzi, "Online anomaly detection using statistical leverage for streaming business process events," in Proc. Int. Conf. Advanced Information Systems Engineering, 2021, pp. 195–210.

[14] D. C. Montgomery, Introduction to Statistical Quality Control, 7th ed. Hoboken, NJ: Wiley, 2013.

[15] S. W. Roberts, "Control chart tests based on geometric moving averages," Technometrics, vol. 1, no. 3, pp. 239–250, 1959.

[16] B. Rosner, "Percentage points for a generalized ESD many-outlier procedure," Technometrics, vol. 25, no. 2, pp. 165–172, 1983.

[17] L. An, A.-J. Tu, X. Liu, et al., "Real-time statistical log anomaly detection with continuous AIOps learning," in Proc. Int. Conf. Software Technologies, 2022, pp. 294–305.

[18] "A controller for anomaly detection, analysis and management for self-adaptive container clusters," International Journal on Advances in Systems and Measurements, vol. 12, no. 3&4, pp. 168–179, 2019.

[19] Fluent Bit Documentation, "About Fluent Bit," [Online]. Available: https://docs.fluentbit.io/