THXInLog: Robust Semantic-Temporal Framework for Log Anomaly Detection in Production Supercomputers

Yuqi Li[§], Liquan Xiao[§], Yongqian Sun[†], Xiuhong Tan[‡], Jinghua Feng[‡]
Yiqiang Li[‡], Lei Tao[†], Tongqing Zhou[§], and Yuan Yuan[§]
[§]*College of Computer Science and Technology, National University of Defense Technology*
liyq@nscc-tj.cn, {xiaoliquan,zhoutongqing,yuanyuan}@nudt.edu.cn
[†]*Nankai University*
sunyongqian@nankai.edu.cn, leitao@mail.nankai.edu.cn
[‡]*National SuperComputer Center in Tianjin*
{tanxh,fengjh,liyiqiang}@nscc-tj.cn

***Abstract*—Production supercomputer logs are massive, heterogeneous, and continuously evolving, limiting methods that rely on labeled data, fragile templates, or costly inference. We propose THXInLog, a template-free framework combining a semantic-temporal detector initially trained without anomaly labels with expert-verified closed-loop adaptation. The detector learns HPC-adapted representations from raw logs and uses a Dual-Scope Transformer to model local event dependencies and window-level operational context. During deployment, an LLM assists alert triage, while accumulated expert-verified corrections are used to periodically update the detector and its dynamic threshold. Evaluations on Tianhe-NG production logs show that THXInLog achieves an F1 score of 0.9082, outperforming representative baselines by 4-11 percentage points, with a per-log latency of 5.82×10^{-4} s. Case studies and long-term deployment observations further demonstrate its early-warning capability and effectiveness in continuous operation.**

***Index Terms*—Log Anomaly Detection; Contrastive Learning; Supercomputer; Temporal Semantics**

I. Introduction

Supercomputers play a pivotal role in advancing science, engineering, and society, yet their exascale capabilities and intricate hardware–software interdependencies pose formidable reliability challenges. The severity of such challenges is illustrated by the 32 million core hours lost to storage failures on the Blue Waters supercomputer in 2018 [1]. Many high-impact failures in production supercomputers involve anomalies whose operational state cannot be directly or continuously measured, such as intermittent faults, cross-subsystem interactions, or context-dependent performance degradations, and thus can only be identified through comprehensive log analysis. Empirical studies further indicate that such log-reliant anomalies occur frequently and require longer recovery, particularly in filesystems and resource management [2], making automated log-based anomaly detection critical for early fault localization, prevention of cascading failures, and minimization of downtime.

*Corresponding author: Yuan Yuan.

However, robust log anomaly detection in production supercomputers remains difficult because workload shifts, software updates, and configuration changes continuously alter high-volume and heterogeneous log streams. **First**, template-based methods, such as DeepLog [3] and LogAnomaly [4], first compress raw messages into discrete template IDs. This discretization not only inherits parsing errors and template drift, but also discards parameter-level evidence, such as component identifiers, error codes, and durations, while treating semantically related templates as unrelated categories. **Second**, parser-free semantic methods, such as NeuralLog [5], preserve raw-text information, yet generic language encoders are not specifically aligned with HPC terminology and operational context. They also lack an explicit mechanism for preventing isolated background messages from contaminating the subsequent sequence context. Consequently, rare benign events and subtle faults may receive ambiguous representations, while conventional sequence classifiers do not explicitly distinguish short-range event transitions from longer-term operational context. **Third**, LLM-based approaches, including LogPrompt [6], LogLM [7], CoLA [8], and LLMeLog [9], provide stronger semantic reasoning but introduce different deployment trade-offs. Direct prompting incurs high token-level latency and sensitivity to prompts and model versions, whereas enrichment or collaboration strategies depend on LLM-generated knowledge whose errors may propagate to downstream detection. Moreover, semantic reasoning alone does not establish a compact temporal normality model that can be continuously updated for large-scale log streams. Existing approaches therefore trade off semantic fidelity, multi-scope temporal modeling, throughput, and adaptability.

To address these limitations, we propose **THXInLog**, a template-free framework combining a semantic-temporal detector initially trained without anomaly labels with expert-gated, LLM-assisted adaptation. Operating on normalized raw messages preserves semantic and parameter-level evidence. Temporal contrastive adaptation aligns representations with

HPC contexts, LOF-guided masking reduces interference from isolated messages, and a Dual-Scope Transformer captures local and global dependencies. During deployment, the LLM assists alert analysis outside the per-log detection path, while only expert-verified corrections update the detector and its dynamic threshold, enabling efficient and reliable long-term adaptation.

Our key contributions are summarized as follows:

(1) We propose THXInLog, a template-free log anomaly detection framework for production supercomputers that combines label-free initial detection with LLM–expert coordinated closed-loop adaptation. THXInLog operates on normalized raw log messages without extracting discrete templates or requiring manually labeled anomalies for initial training.

(2) We develop an HPC-aware semantic-temporal detector that learns robust raw-log representations through temporal contrastive adaptation and introduces a Dual-Scope Transformer to jointly model local event dependencies and global operational context, enabling reliable detection under heterogeneous and evolving logging conditions.

(3) We conduct extensive evaluations on production logs from the Tianhe-NG supercomputer. THXInLog improves F1 scores by 4–11 percentage points over representative baselines while maintaining a per-log inference latency of approximately 5.82×10^{-4} seconds. Long-term observations further demonstrate its effectiveness for distributional adaptation and early failure detection.

II. Related Work

A. Learning Paradigms for Log Anomaly Detection

Log-based anomaly detection has been extensively studied in distributed systems to support fault detection, diagnosis, and reliability analysis. Early approaches rely on rule-based or statistical techniques [10], while supervised and semi-supervised learning methods improve accuracy by leveraging labeled or partially labeled data [3], [11], [12]. In particular, DeepLog [3] learns sequential patterns over discrete log templates, and PLELog [12] reduces labeling cost via probabilistic label estimation. However, these methods depend on log parsing and assumptions about normal behavior, limiting their robustness under changing system conditions. To reduce reliance on labeled data, unsupervised methods, such as PCA-based detectors [13], and self-supervised frameworks based on proxy tasks [14] have also been explored.

Most existing log analysis methods follow a template-based pipeline that first extracts log templates and then models event sequences [3], [4]. While effective, their performance is highly dependent on parsing quality [15]. To address this limitation, recent studies learn semantic representations directly from raw logs using deep learning techniques. NeuralLog [5] and LogBERT [16] leverage pre-trained language models for log understanding, while Transformer-based architectures further improve contextual modeling [17]. Recently, large language models (LLMs) have emerged as a new paradigm for log analysis, including DivLog [18], LogPrompt [6], LogLM [7], CoLA [8], and LLMeLog [9], which leverage prompt-based reasoning, instruction tuning, or semantic enrichment for anomaly detection and representation learning. Despite these advances, balancing semantic expressiveness, computational efficiency, and adaptability remains an open challenge.

B. Log Anomaly Detection for HPC Systems

Compared with distributed systems, research on anomaly detection in HPC systems has received relatively limited attention, while log analysis has traditionally been used for system understanding, failure analysis, and performance optimization [19], [20]. Representative efforts include GUIDE [21] for guided log analysis, LogScan [22] for failure diagnosis from system logs, and Doomsday [23] for characterizing failure behaviors in large-scale supercomputers. More recently, ClusterLog [24] explored log anomaly detection for HPC storage systems through semantic log clustering, while several studies have investigated automated anomaly detection using system telemetry, including online anomaly detection [25], recurrent unsupervised anomaly detection (RUAD) [26], and the node-level framework NodeSentry [27]. These studies demonstrate the growing adoption of machine learning for improving HPC reliability.

However, existing studies mainly focus on benchmark datasets or specific subsystems, leaving robust anomaly detection for evolving production supercomputer logs underexplored.

III. Proposed Method

A. Overview

THXInLog follows a detect-then-adapt design that combines a base detector initially trained without anomaly labels with LLM–expert coordinated adaptation. As shown in Fig. 1, the base detector is initially trained on normalized raw messages without template parsing or anomaly labels. A TTC-adapted encoder reduces the mismatch between generic embeddings and HPC log semantics, while LOF-guided masking limits interference from isolated background messages. The resulting sequences are processed by a Dual-Scope Transformer to capture both local event transitions and global operational context, and reconstruction errors are evaluated using a dynamic threshold. During deployment, the LLM assists only in alert analysis rather than per-log inference, while expert-verified detection errors are used to update the detector and threshold. This design preserves raw-log semantics, supports multi-scope temporal modeling, avoids costly online LLM inference, and adapts to evolving production environments. Operational annotations are used only for offline evaluation, whereas expert-verified corrections collected after deployment are used for controlled adaptation.

B. Semantic Representation Learning

To bridge the semantic gap between general-purpose language models and production supercomputer logs, THXInLog employs Triplet Temporal Contrastive (TTC) learning to adapt a pre-trained sentence encoder. This design is motivated by the observation that messages within the same operational context

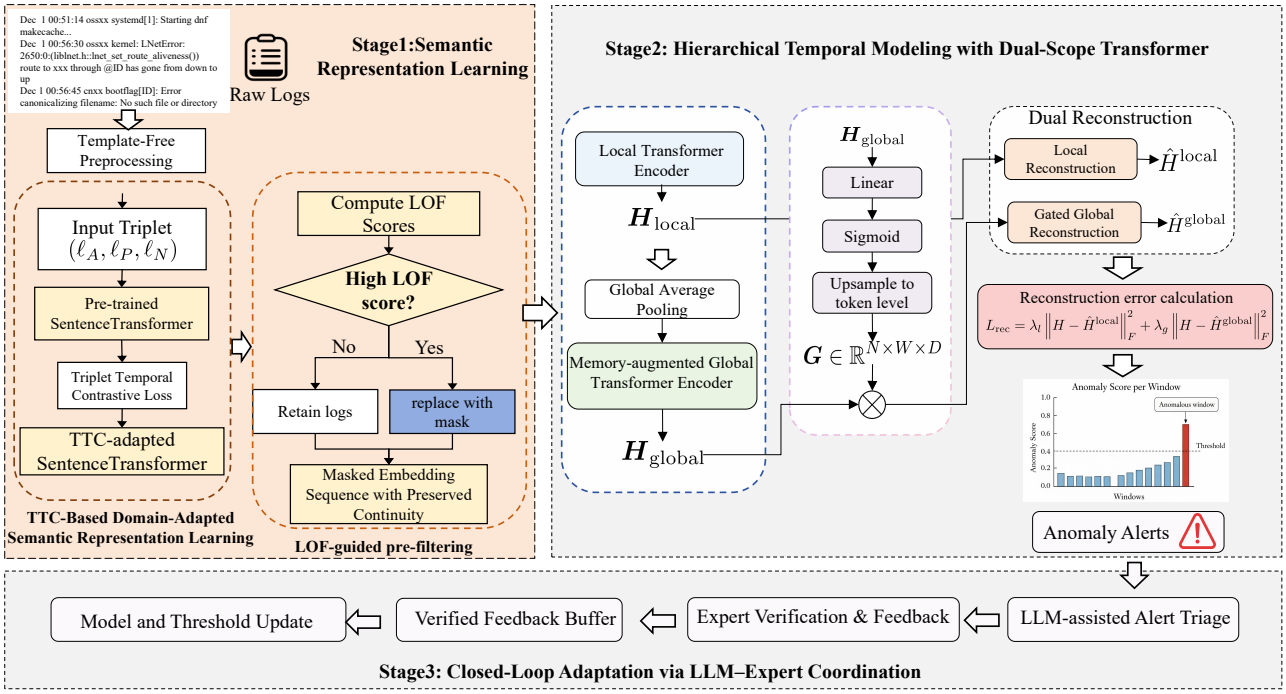

Fig. 1. Overall architecture of THXInLog.

can be lexically dissimilar but behaviorally related, whereas semantically similar messages may have different meanings across contexts. TTC therefore incorporates temporal relations as label-free weak supervision to preserve both semantic proximity and operational consistency.

For each anchor-positive-negative triplet (ℓ_A, ℓ_P, ℓ_N) with embeddings (z_A, z_P, z_N), TTC minimizes

$$\mathcal{L}_{\text{tri}} = \max(0,\, d(z_A, z_P) - d(z_A, z_N) + \alpha)\,, \tag{1}$$

where $d(\cdot,\cdot)$ denotes the embedding distance and α is the margin. Triplets are constructed from source metadata and temporal proximity. Positive samples are temporally adjacent messages sharing the same node and process/component, whereas negative samples are selected from non-overlapping time ranges and different process/components. Requiring both contextual and temporal separation for negative sampling avoids treating recurring periodic messages as unrelated. This construction uses logging metadata as weak supervision without requiring anomaly labels.

Even after domain adaptation, isolated background messages may distort the subsequent temporal context. THXInLog therefore applies LOF-guided masking as a representation-reliability mechanism rather than using LOF directly for anomaly detection. Given a log window $X = \{\ell_1, \ldots, \ell_n\}$, each log message ℓ_i is encoded as z_i by the TTC-adapted encoder, after which its Local Outlier Factor (LOF) score is computed as

$$\text{LOF}_k(z_i) = \frac{1}{|\mathcal{N}_k(z_i)|} \sum_{z_j \in \mathcal{N}_k(z_i)} \frac{\text{lrd}_k(z_j)}{\text{lrd}_k(z_i)}, \tag{2}$$

where $\mathcal{N}_k(z_i)$ denotes the k nearest neighbors of z_i, and $\text{lrd}_k(\cdot)$ is the local reachability density. Messages with LOF scores exceeding a threshold c are marked for masking:

$$L_{\text{mask}} = \{\ell_i \in X \mid \text{LOF}_k(z_i) > c\}. \tag{3}$$

Rather than removing these messages, THXInLog replaces their embeddings with a learnable mask embedding. This preserves the occurrence and position of locally isolated messages while preventing their embeddings from dominating the sequence context, leaving the final anomaly decision to the Dual-Scope Transformer. During deployment, c is updated using expert-verified feedback to accommodate changing logging characteristics.

C. Hierarchical Temporal Modeling with Dual-Scope Transformer

Supercomputer failures often manifest as short event transitions whose interpretation depends on the overall state of the surrounding log window. Unlike conventional stacked Transformers that repeatedly process the same sequence at a single granularity [11], the Dual-Scope Transformer (DST) separates log-entry-level dependencies from window-level operational context and feeds the latter back to local representations.

Log embeddings are organized into sliding windows $\mathbf{H} \in \mathbb{R}^{W \times d}$ and standardized using training-set statistics. DST learns dominant operational patterns from these unlabeled windows. In the *local scope*, positional encoding and self-attention capture dependencies among log entries within each window, producing $\mathbf{X}_{\text{local}}$. In the *global scope*, $\mathbf{X}_{\text{local}}$ is pooled into a window-level representation and refined through

attention over learnable memory slots. The resulting global context is mapped back to the entry level through upsampled sigmoid gating to modulate the local representations.

DST produces a local reconstruction $\hat{\mathbf{H}}^{\text{local}}$ and a gated global reconstruction $\hat{\mathbf{H}}^{\text{global}}$, optimized by

$$\mathcal{L}_{\text{rec}} = \lambda_l \left\| \mathbf{H} - \hat{\mathbf{H}}^{\text{local}} \right\|_F^2 + \lambda_g \left\| \mathbf{H} - \hat{\mathbf{H}}^{\text{global}} \right\|_F^2, \quad (4)$$

where λ_l and λ_g balance the two reconstruction objectives. This dual objective preserves sensitivity to local event deviations while enforcing consistency with window-level operational patterns.

D. Closed-Loop Adaptation via LLM–Expert Coordination

LLMs can efficiently interpret heterogeneous alert logs and retrieve relevant operational experience, but their outputs are not sufficiently reliable to serve directly as training labels. THXInLog therefore assigns the LLM an advisory role and uses expert verification as the gate for adaptation. This separation keeps the LLM outside the per-log detection path while preventing unverified outputs from propagating into the detector.

The base detector is initially trained without anomaly labels. SPOT [28] dynamically maintains an alert threshold z_α over reconstruction errors, and alerts are raised when this threshold is exceeded. For each alert, the LLM summarizes the flagged logs, suggests fault categories, and retrieves similar verified cases; engineers then confirm, reject, or correct its analysis. Only expert-confirmed detection errors are used for adaptation: false positives are added to the normal update set, while missed anomalies are excluded from normal-pattern learning only after confirmation by correlated alerts or post-incident review. When the number of verified corrections accumulated over window T satisfies $N_c > N_{\text{th}}$, the detector and thresholds are updated. N_{th} is set on a predeployment validation period to balance adaptation speed and stability.

IV. EVALUATION

A. Experimental Setup

Dataset and Environment. Experiments were conducted on production logs collected from a service-capable subcluster of the Tianhe-NG supercomputer [29]. Logs were continuously collected through the production logging infrastructure from January to September 2024. Anomaly annotations were obtained from daily operational records and used only for offline evaluation, rather than for initial detector training. Following the monthly operational cycle of the system, logs from January to March were used for training, while logs from April to September were used for evaluation, emulating continuous deployment on future operational data. For the accuracy-efficiency study in TableIV, we further constructed two temporally separated datasets from the same Tianhe-NG subcluster: D1 (688,124 logs, with a higher anomaly ratio) and D2 (139,254 logs, dominated by routine operations).

Baselines and Metrics. We compare THXInLog with representative methods covering different learning paradigms, including statistical methods (PCA [13]), template-based sequence models (DeepLog [3] and PLELog [12]), semantic representation learning (NeuralLog [5]). Detection performance is evaluated using weighted Precision, Recall, and F1 Score to account for the severe class imbalance in production supercomputer logs, with weighted F1 Score reported as the primary evaluation metric unless otherwise specified. For the accuracy–efficiency comparison in Table IV, we further include the LLM-based LogPrompt [6], implemented using Qwen3 [30] under the same hardware configuration.

TABLE I
PERFORMANCE COMPARISON ACROSS DIFFERENT METHODS.

Method	Precision	Recall	F1 Score
NeuralLog	0.9043	0.8139	0.8329
DeepLog	0.9166	0.8431	0.8563
PLELog	0.9173	0.8503	0.8609
PCA	0.7915	0.8150	0.7963
THXInLog	**0.9228**	**0.8943**	**0.9082**

TABLE II
PERFORMANCE OF THXINLOG WITH AND WITHOUT CLOSED-LOOP ADAPTATION.

Variant	Met.	Apr	May	Jun	Jul	Aug	Sep
w/o feedback†	P	0.938	0.898	0.883	0.883	0.893	0.973
	R	0.932	0.911	0.859	0.895	0.756	0.968
	F1	0.934	0.899	0.865	0.887	0.783	0.969
THXInLog	P	0.938	0.910	0.884	0.897	0.944	0.984
	R	0.932	0.922	0.873	0.905	0.920	0.974
	F1	0.934	0.913	0.871	0.897	0.924	0.975

†Without closed-loop dynamic feedback.

B. Performance Evaluation

Table I reports the comparative evaluation of THXInLog against four representative baselines. Among all methods, THXInLog achieves the best overall performance, yielding an F1 score of 0.9082 with a precision of 0.9228 and a recall of 0.8943. THXInLog outperforms NeuralLog by more than 7% in F1 score. Relative to DeepLog and PLELog, THXInLog achieves improvements of about 5% and 4% respectively, while avoiding the reliance on labeled data or fragile templates. Under the same unsupervised setting, THXInLog surpasses PCA by more than 11% in F1 score, demonstrating the effectiveness of its semantic adaptation over linear dimensionality reduction. These results demonstrate that THXInLog consistently delivers superior detection accuracy across different settings and validate the advantage of its semantic adaptation mechanism over both supervised and unsupervised baselines.

Table II compares THXInLog with and without closed-loop adaptation during the six-month prospective evaluation from April to September. Adaptation in each month used only expert-verified feedback available up to that month, without accessing future-month annotations. Even without feedback, THXInLog maintains stable detection performance, indicating

TABLE III
ABLATION STUDY.

Model Configuration	Precision	Recall	F1 Score
TTC + LOF (Full Model)	0.9228	0.8943	0.9082
TTC Only	0.8421	0.7843	0.8651
LOF Only	0.8466	0.8139	0.8678
No TTC, No LOF	0.7934	0.7461	0.8124

TABLE IV
INFERENCE OVERHEAD AND DETECTION PERFORMANCE.

Method	Latency (s/log)	GPU Mem. (%)	F1(D1)	F1(D2)
THXInLog	5.82×10^{-4}	17.76	**0.9435**	**0.9528**
NeuralLog	7.54×10^{-4}	28.13	0.7228	0.7433
PCA	6.87×10^{-7}	0	0.8214	0.8352
DeepLog	4.56×10^{-5}	3.36	0.8563	0.8749
PLELog	3.85×10^{-5}	2.01	0.8859	0.9027
LogPrompt	5.02×10^{-1}	83.24	0.5317	0.7683

that its semantic representation learning and hierarchical temporal modeling are inherently robust to evolving log distributions. Closed-loop adaptation further improves performance in most months, particularly under pronounced distribution shifts. For example, the F1 score increases from 0.783 to 0.924 in August (approximately 18%), while reaching the highest value of 0.975 in September. These results demonstrate that closed-loop adaptation effectively mitigates long-term distribution drift and supports reliable long-term deployment in production supercomputers.

C. Ablation Study

To quantify the contribution of semantic representation learning and log pre-filtering, we evaluate four model configurations: the complete framework, TTC only, LOF only, and a variant without either component. As shown in Table III, the full model achieves the best overall performance, obtaining an F1 score of 0.9082. Removing either TTC or LOF consistently degrades all evaluation metrics, while removing both components results in the largest performance drop, with the F1 score decreasing to 0.8124.

The two modules provide complementary benefits. Removing LOF mainly reduces recall from 0.8943 to 0.7843, indicating that noisy and low-quality log messages negatively affect anomaly coverage. In contrast, removing TTC primarily decreases precision from 0.9228 to 0.8466, suggesting that generic semantic representations are less discriminative for production supercomputer logs. Combining TTC and LOF therefore yields higher-quality semantic representations and cleaner training data, leading to the best overall detection performance.

D. Overhead in Production Supercomputers

Table IV compares THXInLog with representative template-based, semantic-based, and LLM-based methods in terms of detection performance and deployment overhead on two production supercomputer datasets. THXInLog consistently achieves the highest F1 scores (0.9435 on D1 and 0.9528 on D2) while maintaining low inference latency (approximately 5.8×10^{-4} s per log) and GPU memory utilization below 18%.

Among template-based methods, DeepLog and PLELog provide relatively low deployment overhead but remain constrained by template dependence, resulting in lower detection performance. NeuralLog improves semantic modeling through pretrained language models, but incurs higher GPU memory consumption without achieving comparable accuracy. Although LLM-based LogPrompt offers stronger semantic reasoning capability, its inference latency exceeds 0.5 s per log and GPU memory utilization exceeds 82%, making it unsuitable for real-time deployment in production supercomputers. Overall, THXInLog provides the best balance between detection accuracy and deployment efficiency, demonstrating its practicality for continuous online monitoring in large-scale HPC environments.

V. DEPLOYMENT EXPERIENCE

In addition to the monthly evaluation in Table II, THXInLog was deployed on the Tianhe-NG storage subsystem at the end of March 2024 and continuously operated through September 2024. It detected 37 of 38 confirmed failures (97.4%), including all 35 hardware failures and two of three software failures. For the detected events, THXInLog raised alerts an average of 14 minutes and 49 seconds before the existing production monitoring system. Hardware failures were detected nearly 10 minutes earlier on average, while software failures were detected more than two hours earlier. The following Lustre incidents illustrate its detection behavior.

A. Multi-MDT Outage

During a production incident from 17:26 to 17:50, MDT2 first became unreachable at 17:26:14, followed by the sequential disconnection of all three MDTs between 17:26:28 and 17:27:26. Client reconnection messages surged, while OS-layer tasks such as `sysstat-collect` remained normal. Service was gradually restored after 17:40.

Among nine representative syslog entries, THXInLog correctly identified all six fault-related entries and rejected all three benign `systemd` logs. PLELog missed the critical reconnection event; PCA and DeepLog each missed three fault entries, while PCA, NeuralLog, and DeepLog each produced one false positive. NeuralLog also missed part of the sequential fault progression. This case demonstrates the benefit of template-free, dual-scope modeling when faults evolve amid benign OS-level logs.

B. OST Thread Blocking

During a Lustre OST blockage, THXInLog identified abnormal RPC timeouts at 16:13:29. Reconnection failures followed at 16:15:59, and several `ll_ost` threads remained stalled for approximately 200 seconds before crashing at 16:16:43. The production monitoring system raised an alert at 16:17:13, 3 minutes and 44 seconds after THXInLog. This case shows that domain-adapted, dual-scope modeling can detect evolving OST failures before conventional monitoring.

VI. Conclusion

This paper presents THXInLog, a template-free framework that combines a semantic-temporal detector initially trained without anomaly labels with LLM-assisted, expert-verified closed-loop adaptation. TTC-based representation learning adapts raw-log semantics to the HPC domain, while LOF-guided masking reduces interference from isolated background messages. The Dual-Scope Transformer further captures local event dependencies and window-level operational context. Accumulated expert-verified detection errors are periodically used to update the detector and its dynamic threshold. Evaluations on Tianhe-NG production logs demonstrate higher detection accuracy than representative baselines with low inference overhead, while long-term deployment demonstrates its adaptability and early-warning capability. Future work will extend THXInLog to joint log-metric modeling and agent-assisted fault diagnosis and recovery.

Acknowledgment

This work is supported by Jing-Jin-Ji Regional Integrated Environmental Improvement-National Science and Technology Major Project (2025ZD1202200).

References

[1] S. Jha, S. Cui, S. S. Banerjee, T. Xu, J. Enos *et al.*, "Live forensics for hpc systems: A case study on distributed storage systems," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020, pp. 1–16.

[2] F. Cappello, A. Geist, W. Gropp, S. Kale, B. Kramer *et al.*, "Toward exascale resilience: 2014 update," *Supercomputing Frontiers and Innovations*, vol. 1, no. 1, pp. 4–27, 2014.

[3] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1285–1298.

[4] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei *et al.*, "Loganomaly: unsupervised detection of sequential and quantitative anomalies in unstructured logs," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, ser. IJCAI'19. AAAI Press, 2019, p. 4739–4745.

[5] V.-H. Le and H. Zhang, "Log-based anomaly detection without log parsing," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2021, pp. 492–504.

[6] Y. Liu, S. Tao, W. Meng, J. Wang, W. Ma *et al.*, "Interpretable online log analysis using large language models with prompt strategies," in *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension*, ser. ICPC '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 35–46.

[7] Y. Liu, Y. Ji, S. Tao, M. He, W. Meng *et al.*, "Loglm: From task-based to instruction-based automated log analysis," in *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2025, pp. 401–412.

[8] X. Zhu, X. Tang, S. Wu, J. Li, H. Wang *et al.*, "Cola: Model collaboration for log-based anomaly detection," *Proc. VLDB Endow.*, vol. 18, no. 11, p. 3979–3987, Jul. 2025.

[9] M. He, T. Jia, C. Duan, H. Cai, Y. Li *et al.*, "Llmelog: An approach for anomaly detection based on llm-enriched log events," in *2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE)*, 2024, pp. 132–143.

[10] X. Fu, R. Ren, S. A. McKee, J. Zhan, and N. Sun, "Digging deeper into cluster system logs for failure prediction and root cause diagnosis," in *2014 IEEE International Conference on Cluster Computing (CLUSTER)*, 2014, pp. 103–112.

[11] S. Huang, Y. Liu, C. Fung, R. He, Y. Zhao *et al.*, "Hitanomaly: Hierarchical transformers for anomaly detection in system log," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2064–2076, 2020.

[12] L. Yang, J. Chen, Z. Wang, W. Wang, J. Jiang *et al.*, "Plelog: Semi-supervised log-based anomaly detection via probabilistic label estimation," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 2021, pp. 230–231.

[13] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, ser. SOSP '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 117–132.

[14] X. Tan, N. Han, S. Lu, W. Chen, and D. Wang, "Semlog: A semantics-based approach for anomaly detection in big data system logs," in *2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS)*, 2023, pp. 1199–1206.

[15] Y. Fu, M. Yan, Z. Xu, X. Xia, X. Zhang *et al.*, "An empirical study of the impact of log parsers on the performance of log-based anomaly detection," *Empirical Softw. Engg.*, vol. 28, no. 1, Nov. 2022.

[16] H. Guo, S. Yuan, and X. Wu, "Logbert: Log anomaly detection via bert," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.

[17] S. Tao, Y. Liu, W. Meng, Z. Ren, H. Yang *et al.*, "Biglog: Unsupervised large-scale pre-training for a unified log representation," in *2023 IEEE/ACM 31st International Symposium on Quality of Service (IWQoS)*, 2023, pp. 1–11.

[18] J. Xu, R. Yang, Y. Huo, C. Zhang, and P. He, "Divlog: Log parsing with prompt enhanced in-context learning," in *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*, 2024, pp. 2457–2468.

[19] A. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, 2007, pp. 575–584.

[20] Z. Zheng, L. Yu, W. Tang, Z. Lan, R. Gupta *et al.*, "Co-analysis of ras log and job log on blue gene/p," in *2011 IEEE International Parallel & Distributed Processing Symposium*, 2011, pp. 840–851.

[21] S. S. Vazhkudai, R. Miller, D. Tiwari, C. Zimmer, F. Wang *et al.*, "Guide: A scalable information directory service to collect, federate, and analyze logs for operational insights into a leadership hpc facility," in *SC17: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2017, pp. 1–12.

[22] B. H. Park, Y. Hui, S. Boehm, R. A. Ashraf, C. Layton *et al.*, "A big data analytics framework for hpc log data: Three case studies using the titan supercomputer log," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, 2018, pp. 571–579.

[23] A. Das, F. Mueller, P. Hargrove, E. Roman, and S. Baden, "Doomsday: Predicting which node will fail when on supercomputers," in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2018, pp. 108–121.

[24] C. Egersdoerfer, D. Zhang, and D. Dai, " ClusterLog: Clustering Logs for Effective Log-based Anomaly Detection ," in *2022 IEEE/ACM 12th Workshop on Fault Tolerance for HPC at eXtreme Scale (FTXS)*. Los Alamitos, CA, USA: IEEE Computer Society, Nov. 2022, pp. 1–10.

[25] A. Borghesi, A. Libri, L. Benini, and A. Bartolini, "Online anomaly detection in hpc systems," *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 229–233, 2019.

[26] M. Molan, A. Borghesi, D. Cesarini, L. Benini, and A. Bartolini, "Ruad: Unsupervised anomaly detection in hpc systems," *Future Generation Computer Systems*, vol. 141, pp. 542–554, 2023.

[27] S. Xia, Y. Sun, X. Pan, Y. Yuan, S. Zhang *et al.*, "Effective node-level anomaly detection in hpc systems via coarse-grained clustering and fine-grained model sharing," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 1180–1194.

[28] A. Siffer, P.-A. Fouque, A. Termier, and C. Largouet, "Anomaly detection in streams with extreme value theory," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1067–1075.

[29] Y. Dai, Y. Dong, K. Lu, R. Wang, W. Zhang *et al.*, "Towards scalable resource management for supercomputers," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '22. IEEE Press, 2022.

[30] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui *et al.*, "Qwen3 technical report," 2025. [Online]. Available: https://arxiv.org/abs/2505.09388