FLMLog: A Federated LLM Framework for Unified Online Log Anomaly Detection

Shuai Xu[1][0000-0003-3897-468X] , Guanghong Yang[2], Zepeng Wen*

Institute of Computer Application
China Academy of Engineering Physics, Mianyang, China
sa615527@mail.ustc.edu.cn
{1245122413,280110270}@qq.com

Abstract. Effective anomaly detection is essential for guaranteeing the stable and reliable operation of contemporary large-scale distributed computing systems. Traditional centralized log anomaly detection frameworks have inherent flaws, as they are prone to privacy leakage risks when transmitting system data across networks. Previous research mainly focuses on single-domain logs, requiring domain-specific models and retraining, which limits flexibility and scalability. While LLMs have made great strides in natural language understanding and text generation, they are constrained by excessive computational costs and limited training data availability. Federated learning effectively addresses these issues by enabling decentralized LLM training, which makes full use of scattered local data while avoiding private data leakage. Such privacy-preserving features render federated LLMs well-suited for sensitive industrial scenarios. Accordingly, this paper constructs FLMLog, a holistic online log anomaly detection framework that integrates federated learning and LLM techniques. To enhance the operational efficiency, the FLMLog framework adopts a prefix-aware in-context learning (ICL) refinement strategy. This strategy is specifically designed to refine both the selection of in-context examples and the permutation order of these examples, thereby achieving an improvement in prefix caching efficiency. We evaluate the proposed FLMLog framework on five public industrial log datasets. Experimental results demonstrate its superior overall detection performance. In comparison with advanced baseline methods, FLMLog yields consistent F_1-score improvements across four datasets, showing robust and competitive detection performance.

Keywords: Anomaly Detection, Federated Learning, LLMs.

1 Introduction

With the gradual expansion of enterprise business operations, an increasing number of modern applications are evolving toward large-scale and highly complex architectural paradigms. In this context, the distributed microservices architecture stands out for its

* Corresponding author.

ability to enable independent development, deployment, upgrade, and scaling of individual services. Distributed microservice architectures have become a mainstream software design paradigm in modern industrial development, benefiting from their prominent strengths including modular design, independent deployment, reusable components and flexible elastic scaling. Nevertheless, the large-scale deployment of microservice systems also brings new critical reliability risks to system operation. System failures can severely degrade user satisfaction and cause substantial business losses; industry analyses estimate that the average hourly cost of system downtime ranges from $301,000 to $400,000 [11], high-lighting the urgent need for robust fault tolerance and operational resilience. Systems require early anomaly monitoring and accurate root error pinpointing to maintain stable performance, where full records of running data provide major support for abnormality analysis. This practice has rendered log-based anomaly detection [9] a critical research domain with significant industrial and academic value.

Traditional manual and rule-based log detection methods, which rely on domain knowledge for multi-pattern matching and keyword searches, face inherent limitations in large-scale distributed systems due to three core challenges: label noise contamination from mixed normal and anomalous training samples [2], exponentially growing log volumes (reaching gigabytes to terabytes per hour)[4], and heterogeneous log schemas caused by proprietary vendor standards [16]. Additionally, with the rising number of users and Quality of Service (QoS) demands, log data volume surges-conservative estimates show a 1,000-employee company generates over 100 GB of daily log traffic, with peak events per second exceeding 22,000 [13] and false alerts remain prevalent (only 24.1% of security alerts are major threats [15]), further complicating anomaly identification.

Deep learning methods have been widely applied to address these issues, leveraging their strengths in time-series processing and long-range dependency capture. Nevertheless, existing models suffer from centralized training's bandwidth consumption and privacy risks [3], as well as poor adaptability to multi-domain logs. Although recent studies (e.g., Guo et al. 2024 [17]) have explored cross-domain learning, most still focus on single-domain detection, limiting practical applicability. In industrial scenarios with multiple log sources (e.g., data center storage and super computing logs), cross-domain anomaly detection is promising but faces data heterogeneity and Non-IID data challenges in federated learning environments, where Transformer-based models are prone to overfitting and gradient conflicts during global aggregation, reducing overall effectiveness.

Targeting the above-mentioned technical dilemmas, this work presents FLMLog, a unified online log anomaly detection framework built upon federated learning (FL) and large language models (LLMs). We first explore the feasibility of applying federated learning to log anomaly detection based on deep learning techniques. Different from traditional AI models that require centralized data collection, federated learning builds a global model at the central server by merely exchanging locally optimized model parameters, without accessing original log records. Such a decentralized training paradigm effectively avoids the exposure of sensitive log information and guarantees overall data privacy and security. Furthermore, FL distributes training workloads to diverse

participant nodes, enabling raw log data to be retained locally rather than gathered on a centralized server. This distributed design not only avoids the bandwidth overhead associated with the transmission of massive log data but also significantly improves the efficiency of data processing and model updates, which is crucial for adapting to the large-scale and dynamic characteristics of modern distributed microservice systems.

First, to achieve efficient log file parsing, a large language model (LLM) inference optimization method is designed for online log parsing. In the proposed framework, each federated client builds an individual LLM to implement real-time log parsing and extract critical feature information from parsed log content. This procedure generates high-quality feature data for subsequent model optimization, supporting the stable construction of local anomaly detection models on all client nodes. After finishing local training, every node uploads its optimized model parameters to the central server. The server integrates all received local parameters via a standardized aggregation mechanism to generate a unified global detection model. The resulting global model possesses strong generalization capability, enabling it to adapt to cross-domain log anomaly detection tasks.

To validate the effectiveness of the proposed approach, we first elaborate on the federated large language models (LLMs) tailored for unified online log anomaly detection, explicitly specifying the detailed detection steps and standard operational procedures. For efficient online log parsing, a dedicated LLM inference optimization method is adopted. At the local node level, each server trains a local anomaly detection model based on a self-attention-based bidirectional long short-term memory (BiLSTM) architecture, which is capable of capturing complex temporal patterns and anomalous behaviors embedded in log sequences. Subsequently, the central server performs parameter aggregation on the trained local model parameters, thereby updating the global anomaly detection model and ensuring its consistency, generalization, and effectiveness across multiple federated nodes.

Experiments conducted on the HDFS log dataset verify the efficacy of the proposed approach. Comparative analyses against prevailing existing methods prove that the designed log parser achieves remarkable efficiency improvements. Following feature extraction and model training procedures, the framework sustains an anomaly detection accuracy above 93%. The federated structural design successfully avoids private data exposure while retaining excellent detection performance, which validates the model's superior stability and robustness. Extensive evaluations on multiple datasets further confirm the universal effectiveness and broad applicability of our method, excluding accidental experimental results. The key innovations and contributions of this study are outlined as follows:

(1) This paper develops FLMLog, an innovative unified online log anomaly detection framework that integrates the strengths of federated learning and large language models. This novel method effectively mitigates two common drawbacks of conventional LLM-based solutions, namely inferior data quality adaptability and excessive computational resource consumption;

(2) The FLMLog framework adopts a prefix-aware in-context learning (ICL) refinement strategy to enhance operational efficiency;

(3) Two microservice test systems with 10 and 28 running instances are utilized to assess the practical performance of the federated learning-driven framework. Quantitative experimental results reveal that our method obtains F_1-scores of 0.793 and 0.835 on the two test platforms. In comparison with mainstream baseline approaches, the proposed framework delivers average performance improvements of 0.189 and 0.195 in F_1 metric, demonstrating its superior detection superiority.

The rest of this paper is structured in the following manner. Section II reviews the research background and existing studies concerning log-based anomaly detection. Section III elaborates on the detailed implementation of the federated learning-enabled framework for log anomaly detection. Section IV presents the experimental setup and comprehensive evaluation results. Lastly, Section V summarizes this work and draws the conclusion.

2 Problem Definition

Log anomaly detection is formally defined as a binary classification problem. Given an input log sequence, the model is tasked with determining whether the sequence is normal or anomalous. In the multi-source log federated learning framework, the log data involves M categories, with each category distributed across different clients. Let the total number of clients be denoted as T, and the local dataset of each client t (where $t \in \{1, 2, \ldots, T\}$) be represented as DC_t, where $DC_t = \{(a_i, b_i)\}^{mt}_{i=1}$. Here, a_i denotes the log sequence, $b_i \in \{0, 1\}$ represents the label of the log sequence (with 0 indicating a normal sequence and 1 indicating an anomalous sequence), and m_t denotes the size of the local dataset of client t. The local dataset DC_t of each client t can be further represented as:

$$DC_t = \bigcup_{c=1}^{M} DC_t^{(c)} \tag{1}$$

where $DC_t^{(c)}$ represents the subset of the dataset on client t that belongs to class c, and $DC_t^{(c)} = \{(a_i, b_i) \mid a_i \in class\ c\}$. In each round of federated learning, the central server randomly selects N clients from the total of T clients to participate in the training process. Let the set of selected clients be denoted as SC, where $|SC| = N$. Each selected client $t \in SC$ trains the model on its local dataset DC_t, and obtains the local model parameters θ_t. The training process can be expressed as:

$$\theta_t = arg\ min \sum_{(a_i, b_i \in DC_t)} \mathcal{L}(f(a_i; \theta), b_i) \tag{2}$$

where $f(a_i ; \theta)$ denotes the model's prediction for the input a_i, and $\mathcal{L}(\cdot)$ represents the loss function (e.g., cross-entropy loss).

3 Related Work

3.1 Log Parsing

Log parsing aims to transform semi-structured log records into standardized structured data. It extracts fixed log templates as static elements and changeable log parameters as variable components from raw log information. Current log parsing techniques are generally classified into two major types, including syntax-dependent and semantics-driven parsing methods. Syntax-based log parsers [19] typically adopt manually designed heuristics or analyze syntactic characteristics to extract log templates. Nevertheless, these tools suffer from degraded precision when handling log data that deviates from standard formatting rules. In contrast, semantic-based log parsers [5] leverage neural networks or language models to identify log templates and parameters by understanding the underlying semantics of log messages. For instance, LogPPT [21] employs a RoBERTa model to recognize log templates and parameters through few-shot learning. In recent years, the rapid advancement of large language models has facilitated the innovation of novel LLM-powered log parsing tools. Such approaches deliver superior parsing performance and better generalization ability for heterogeneous log formats. These LLM-based parsers adopt techniques such as fine-tuning [23] and prompt engineering [26] to optimize LLMs specifically for log parsing tasks, thereby achieving remarkable performance.

3.2 Log Anomaly Detection

Advanced methodologies adhere to a standardized workflow comprising log parsing, feature extraction, and anomaly detection [5]. Deep learning-based approaches capture complex patterns in normal execution logs (e.g., sequential dependencies, quantitative relationships) and flag sequences exhibiting deviations from these learned normal patterns [3]. [16] explicitly incorporates semantic relationships among words, logs, and sequences. This existing method constructs a hierarchical word-log-sequence structure and develops a log anomaly detection framework based on multi-level semantic log representations. Its key innovation combines probabilistic label estimation with historical anomaly patterns, successfully introducing the strengths of supervised learning into conventional machine learning workflows. LOGLG [10] introduces a novel weakly-supervised framework for log anomaly detection to explore semantic correlations between keywords within sequences. Current anomaly detection approaches built on centralized learning mainly prioritize log text processing and model prediction accuracy, while overlooking two vital indicators: data privacy protection and system operational robustness.

3.3 Federated Learning

Under the federated training framework, clients train a shared global model through edge-cloud communication, where a central server is responsible for client selection, weight distribution, and result aggregation. Common parameter aggregation algorithms include FedProx [23] and SCAFFOLD [14]. Unlike traditional aggregation strategies, frameworks such as FedSym [18] adopt a model symbiosis approach, enabling client

models to collaborate through parameter fusion and layer recombination. Cross-domain log anomaly detection is predominantly hindered by data heterogeneity issues. Specifically, log data collected from diverse client nodes usually presents non-independent and identically distributed (Non-IID) characteristics. This Non-IID characteristic leads to discrepancies in data features, labels, or data quality. For instance, logs generated by storage servers and computing servers differ significantly in terms of task types, data volume, log formats, and temporal dynamics, which complicates the training of a cross-domain log anomaly detection model. Additionally, issues such as data imbalance and label class imbalance within a single domain can further degrade model performance. In the field of log anomaly detection, Guo et al. (2021) proposed FLOGCNN [20], a lightweight federated learning model designed for single-domain log anomaly detection. This work demonstrates the potential of federated learning in log anomaly detection scenarios. However, FLOGCNN's reliance on a single log dataset restricts its practical applicability, and its performance degrades significantly when confronted with data heterogeneity in multi-domain environments.

4 Approach

4.1 Federated LLM Framework

As visualized in Fig.1, this study presents FLMLog, an end-to-end online log anomaly detection framework integrated with federated learning (FL) and large language models (LLMs). The overall architecture of FLMLog comprises two core modules: the local model training layer and the global parameter aggregation layer. The training layer adopts the architectural design of the Hadoop Distributed File System (HDFS), which inherits its outstanding capabilities in distributed data storage and efficient data management. In this layer, each individual node independently conducts data preprocessing and local model training, ensuring the autonomy and privacy of distributed data.

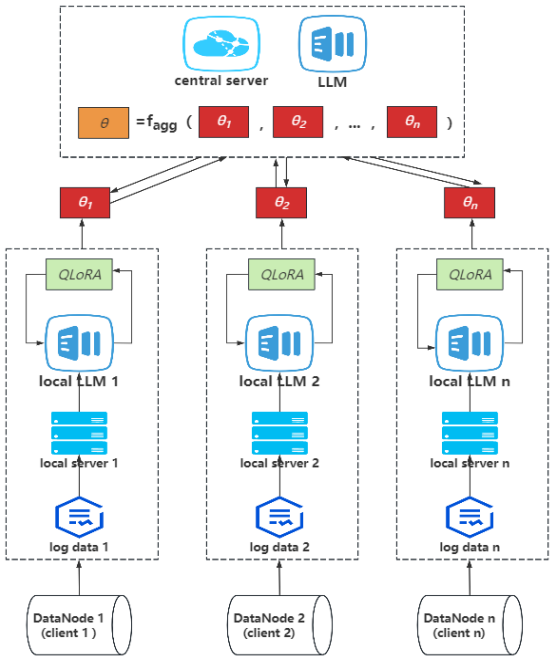

Fig. 1. Federated Learning-based Anomaly Log Detection Architecture

Each DataNode is responsible for local storage maintenance and data block storage operations. By independently storing and processing local log files, it effectively eliminates privacy exposure risks caused by centralized data collection. In contrast, the NameNode undertakes data block management tasks, tracking the storage locations and access routes of all data blocks. It also integrates a write-ahead logging (WAL) mechanism, which records all data alteration behaviors in advance before formal data submission. In this framework, the central server collects locally optimized model parameters uploaded by each DataNode. The global model is iteratively updated via a federated averaging-based BiLSTM optimization strategy. The aggregated global parameters are subsequently distributed to all DataNodes to support successive training iterations. Through decentralized local model training and centralized parameter aggregation, the proposed system possesses multiple prominent technical merits as follows:

(1) **Enhanced Privacy Preservation**: All raw data processing is performed on the client side only, which avoids privacy leakage of confidential data in the course of network interaction and data delivery;

(2) **Efficient Distributed Computation**: This framework makes full use of the computing capacity of each local node, rationally allocates computational tasks across participants, and effectively accelerates the overall model training process;

(3) **Global Model Consistency**: Global parameters are aggregated at the central node, which guarantees the stability and overall effectiveness of the shared global model.

In federated learning scenarios with n participating client nodes, a total of n groups of model parameters $\{\theta_1, \theta_2, \ldots, \theta_n\}$ can be acquired after each training iteration. The

geometric median refers to the optimal data point that minimizes the sum of Euclidean distances to all other samples in a given dataset. Accordingly, the core objective of geometric median-based aggregation is to solve for the global model update that achieves the minimum total distance between all local parameter sets and the final aggregated global parameter point:

$$f_{agg} = \arg\min \sum_{k=1}^{n} ||\theta_k - \theta|| \tag{3}$$

In the formula, $||\cdot||$ stands for the Euclidean norm. Direct calculation of the geometric median involves extremely high computational overhead, so iterative optimization approaches are widely adopted to approximate the optimal solution. The Weiszfeld algorithm serves as an efficient iterative solver for this task, and its specific update rule is defined as follows:

$$\theta^{(t+1)} = \frac{\sum_{k=1}^{n} \frac{\theta_k}{||\theta_k - \theta^{(t)}||}}{\sum_{k=1}^{n} \frac{1}{||\theta_k - \theta^{(t)}||}} \tag{4}$$

By continuously iterating and optimizing the parameter $\theta^{(t)}$ until convergence, this algorithm is capable of yielding an approximate geometric median solution.

4.2 Federated Learning-based Approach

Federated learning adopts a privacy-preserving training paradigm, where participating clients conduct standalone local model training instead of sharing original training data. Only optimized model updates are delivered to the central node for global aggregation. The specific federated workflow tailored for log anomaly detection is illustrated as follows:

(1) Every NameNode is equipped with an independent local server to conduct training with its exclusive local data resources;

(2) A central global server is set up to integrate model weight information uploaded by all local nodes;

(3) A unified deep neural network architecture is deployed on every local server instance;

(4) Each local node executes iterative training relying on feature vectors extracted from local log records in line with preset training rounds;

(5) Upon finishing local iterative training, each local node uploads its optimized weight parameters to the global server. Importantly, original log data will never be delivered to the central server. This design blocks raw data transmission channels and fully safeguards data security in the communication process.

Consistent LLM architectures are adopted for both client and server terminals in the built system. Considering the constrained computing resources on client devices, Parameter Efficient Fine-Tuning (PEFT) is applied for local model optimization. In the federated LLM training workflow, the central server first distributes the initial model parameters θ to all participating clients. Each client subsequently fine-tunes the received parameters with private local datasets and uploads the updated parameter θ_k

back to the server. Finally, the server integrates all received local parameter variations to accomplish one complete training cycle.

Integrating the self-attention module into the BiLSTM-based log anomaly detection model allows weighted fusion processing for log sequence data. By assigning differentiated attention coefficients to log text information, the model can emphasize the critical feature tokens that dominate anomaly discrimination. This optimization enables the network to effectively mine deep latent features from log data and capture the variation rules of log sequences triggered by abnormal behaviors. As a result, the overall detection capability of the model is significantly improved, supporting timely and accurate early warning of log anomalies. The detailed workflow of embedding the self-attention mechanism into text feature learning is presented below.

(1) The input text matrix is linearly transformed through three independent weight matrices to generate three feature matrices, namely Q_B, K_B, and V_B ;

(2) The Softmax activation function is adopted to convert the normalized weight matrix calculated as $W = Softmax(Q_B K_B/\sqrt{d})$ into the ultimate weight matrix, where the variable $\sqrt{d}$ represents the dimension of the K_B matrix;

(3) The final weighted summation matrix can be derived through the following formula:

$$Attention(Q_B, K_B, V_B) = W * V_B \tag{5}$$

4.3 Using LLMs for Parsing Logs

Traditional fine-tuning strategies for large language models (LLMs) demand massive computing resources and abundant high-quality annotated data. Additionally, software systems are constantly updated, resulting in continuous variations in log templates and greatly limiting the practicability of conventional fine-tuning schemes. As an alternative solution, in-context learning (ICL) has become a prevalent paradigm that enables LLMs to handle downstream tasks without parameter fine-tuning. This technique generates predictive results by relying on task contexts supplemented with a few exemplar samples, which are selected according to their similarity to the target log query. Such a mechanism allows LLMs to rapidly adapt to emerging log formats and new template patterns, effectively lowering the model's reliance on labeled training data. Each exemplar sample adopted in this work is composed of a well-labeled pair of raw log data and its corresponding log template. For individual log parsing tasks, the following prompt format has been widely adopted: Common Instruction: <...>, Demonstration Examples: <...>, Queried Log: <...>.

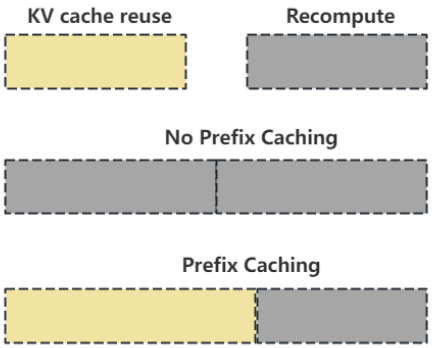

Fig. 2. Comparison of Prefix Caching

The prefill phase constitutes a major computational bottleneck, as it requires generating key (K) and value (V) tensors for the entire input sequence. Benefiting from the autoregressive generation mechanism of large language models, each newly produced token relies exclusively on previously generated outputs. To accelerate inference, attention-derived KV tensors can be cached; during subsequent forward propagation, the model only takes the latest token and cached KV states as inputs rather than recalculating the entire sequence. Prefix caching (PC) serves as an effective optimization strategy to enable cross-request reuse of KV cache resources. This technique computes the shared prompt context only once and allows all queries with identical prefix segments to reuse the cached results. In practice, new requests with matching prefixes can directly inherit existing KV cache data and skip redundant computation for overlapping token sequences. Restricted by limited hardware memory capacity, cached KV blocks are eliminated following the least recently used (LRU) replacement principle, as demonstrated in Fig.2.

Taking Fig.3 as an example, three requests arrive in sequence. Requests 2 (req2) and 3 (req3) share part of the prefix with request 1 (req1), thus they can load the *KV* cache without recomputation (i.e., [b1, b2] for req2 and [b1, b2, b3] for req3).

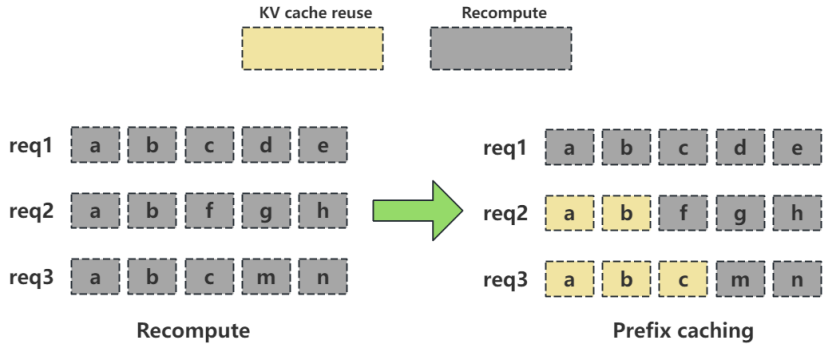

Fig. 3. The comparison of recompute and PC

4.4 Large language model training

Anomaly detection is a binary task with normal and abnormal categories, whose sample sizes are unknown. We use minority oversampling to mitigate data imbalance and keep the minority ratio above β. Let α denote the original minority proportion with $\alpha < \beta$, and sample_num the total dataset size. The required augmented minority count to hit ratio β is derived as follows:

$$\frac{\beta（1-\alpha）}{1-\beta} \times sample\_num \tag{6}$$

This adjustment calibrates the minority class ratio to exactly β. We tune the model for standardized prediction outputs: it returns "The sequence is anomalous" for abnormal input sequences and "The sequence is normal" for regular ones. Cross-entropy loss is selected as the model's loss metric. To cut the overhead of fine-tuning large parameterized LLMs, QLoRA technology is adopted to lower memory consumption. QLoRA delivers this optimization by conducting gradient backpropagation on a fixed 4-bit quantized base model, without sacrificing the prediction accuracy obtained from complete 16-bit fine-tuning.

5 Evaluation

This experiment relies on a high-performance GPU server cluster as its computational hardware infrastructure. Each node within the cluster is equipped with dual CPUs and 512 GB physical memory, alongside a pair of graphics cards featuring 64 GB HBM2e video memory apiece. Ubuntu 22.04 is adopted as the operating system, while the FLMLog framework is fully implemented using Python 3.14.

For in-context learning (ICL), we select 8 labeled log samples as demonstrations for each prompt input, including 4 normal log samples and 4 anomaly samples. A fixed random seed is adopted to ensure the consistency and reproducibility of sample selection in all experimental runs.

For federated learning, the total number of global communication rounds is set to 50. In each round, 80% of clients are randomly selected for local training. The number of local epochs on each client is 3. The client selection follows a data-volume-weighted random strategy, which prioritizes clients with more balanced data scale and representative distribution, thereby improving the convergence stability under non-IID log data distributions.

For the large language model backbone, we use Qwen2.5-14B as the fundamental large language model. Boasting compact model size and outstanding comprehensive capabilities, this lightweight yet high-performance model achieves an excellent trade-off between inference efficiency and task accuracy across diverse scenarios. For parameter-efficient fine-tuning, we apply QLoRA with rank = 8, LoRA alpha = 16, and dropout rate = 0.1. The model weights are quantized to 4-bit NormalFloat (NF4) with double quantization, and all activations are stored in FP16 for efficiency.

The local self-attention BiLSTM module has a hidden dimension of 256 and 4

attention heads to capture sequential and semantic dependencies in log sequences. For federated aggregation, we use the geometric median method with a convergence threshold $\varepsilon = 1\times10^{-5}$. The maximum log sequence length is truncated or padded to 512. The optimizer is AdamW with an initial learning rate of 2×10^{-5} and weight decay of 1×10^{-6}.

5.1 Datasets

We evaluated the effectiveness of FLMLog on five publicly available datasets, namely HDFS[1], Blue Gene/L (BGL[2]), ThunderBird[3], Spirit[4], and Liberty[5] [39]. Specifically, the HDFS dataset was collected from a Hadoop Distributed File System (HDFS) deployed on the Amazon Elastic Compute Cloud (EC2) platform. The BGL dataset was obtained from the Blue Gene/L supercomputer located at Lawrence Livermore National Laboratory (LLNL). In contrast, the ThunderBird, Spirit, and Liberty datasets were collected from two real-world supercomputers operated by Sandia National Laboratories (SNL). Given the large scale of these three datasets (ThunderBird, Spirit, and Liberty), a random sampling strategy was adopted to ensure the feasibility and efficiency of the evaluation: 5,000,000 log entries were randomly selected from each of the three datasets for subsequent experimental evaluations. The detailed information of each dataset, including their data sources, total number of log entries, and the quantity of anomalous log entries, is summarized in Table 1.

Table 1. Detailed information of the datasets.

Dataset	Category	Messages	Anomalies
HDFS	Distributed system	11175629	16838
BGL	Supercomputer	4747963	348460
ThunderBird	Supercomputer	5000000	76130
Spirit	Supercomputer	5000000	764890
Liberty	Supercomputer	5000000	1814386

5.2 Evaluation Metrics

This study assesses the performance of the aforementioned approaches using commonly employed Precision(*Prec*), Recall(*Rec*) and F_1-score(F_1). These metrics are calculated as follows:

$$Prec = \frac{TP}{TP+FP} \tag{7}$$

[1] https://github.com/logpai/loghub/tree/master/HDFS
[2] https://github.com/logpai/loghub/tree/master/BGL
[3] https://github.com/logpai/loghub/tree/master/Thunderbird
[4] https://github.com/logpai/loghub/tree/master/Spirit
[5] https://github.com/logpai/loghub/tree/master/Liberty

$$Rec = \frac{TP}{TP+FN} \tag{8}$$

$$F_1 = \frac{2*Prec*Rec}{Prec+Rec} \tag{9}$$

, where *TP*, *FN*, *FP* stand for true positives, false negatives, and false positives separately. Precision quantifies the proportion of accurately detected abnormal samples within all model-predicted anomalies. In contrast, recall calculates the proportion of successfully recognized anomalies out of all actual abnormal cases. As a comprehensive evaluation indicator, the F_1-score integrates precision and recall to deliver a holistic and balanced evaluation of the model's anomaly detection capability.

5.3 Results and analysis

When evaluating the performance of FLMLog on a specific dataset, this dataset was designated as the target set, while the remaining four datasets were employed as the support set for constructing the knowledge base. In practice, only a single construction of the meta-learner is required, followed by masking the rows corresponding to the target set in the matrix. For the target set, 10% of the normal logs were randomly selected for model training, whereas the remaining normal logs and all abnormal logs were used for anomaly detection. To mitigate the impact of randomness, three independent experiments were conducted starting from the construction of the meta-learner, and the average value of the results was taken as the final performance metric. To comprehensively validate the effectiveness of FLMLog, its performance was compared against six state-of-the-art unsupervised log anomaly detection baselines, including LogGPT [22], CNN [19], DeepLog [8], LogTAD [12], and LogReT [4].

Table 2. Prec, Rec, and F_1 of different methods on different datasets.

Dataset	Metrics	LogGPT	CNN	DeepLog	LogTAD	LogReT	**FLMLog**
HDFS	*Prec*	0.841	0.701	0.921	0.779	0.957	**0.985**
	Rec	0.936	0.992	0.952	0.928	0.989	**0.998**
	F_1	0.897	0.819	0.940	0.857	0.948	**0.991**
BGL	*Prec*	0.642	0.709	0.688	0.726	0.811	**0.862**
	Rec	0.813	0.652	0.889	0.579	**0.974**	0.923
	F_1	0.720	0.681	0.781	0.641	0.872	**0.897**
Thun-derBird	*Prec*	0.621	0.798	0.637	0.102	0.714	**0.782**
	Rec	0.890	0.689	0.841	0.241	0.912	**0.990**
	F_1	0.729	0.737	0.728	0.125	0.798	**0.868**
Spirit	*Prec*	0.601	0.819	0.655	0.713	0.710	**0.882**
	Rec	0.841	0.651	0.781	0.844	0.898	**0.947**
	F_1	0.689	0.727	0.701	0.771	0.812	**0.921**
Liberty	*Prec*	0.641	0.540	0.789	**0.901**	0.691	0.749

Rec	0.911	0.921	0.829	**0.985**	0.928	0.981
F_1	0.762	0.683	0.816	**0.927**	0.783	0.879

The experimental results, as shown in Table 2, FLMLog exhibits overall superior performance. On HDFS, FLMLog outperforms all competitors across all metrics, achieving *Prec*=0.985, *Rec*=0.998, and F_1=0.991—4.5% higher in F_1 than the second-best LogReT. On BGL, FLMLog leads in *Prec*(0.862) and F_1(0.897), with 2.9% higher F_1 than LogReT, balancing precision and recall despite LogReT's slightly higher *Rec*. For the complex ThunderBird dataset, FLMLog achieves the highest *Rec*(0.990) and F_1(0.868), improving 8.8% in F_1 over LogReT, and its *Prec*(0.782) is significantly higher than most methods. On Spirit, FLMLog excels in all metrics (*Prec*=0.882, *Rec*=0.947, F_1=0.921), with a 26.7% F_1 improvement over CNN. Only on Liberty does FLMLog trail Log-TAD but remains competitive, outperforming four comparison methods. Over-all, FLMLog outperforms the five methods in F_1-score on four datasets, demonstrating strong robustness and adaptability.

Table 3. Comparative evaluation of centralized and federated learning models.

Model	*Prec*	*Rec*	F_1
Centralized BiLSTM	0.961	0.971	0.961
Federated BiLSTM	0.953	0.957	0.952

To explore the performance gaps between the federated learning modeling scheme proposed in this work and conventional centralized learning, comparative experiments are carried out under consistent experimental settings. Table 3 illustrates the quantitative performance comparison between the federated framework and centralized learning methods. The statistical results in Table 3 reveal that the proposed method is slightly inferior to the centralized anomaly detection strategy, with all evaluation indicator fluctuations controlled within 2%. This demonstrates that the federated learning framework achieves competitive anomaly detection performance with only minor accuracy loss, while effectively avoiding potential data privacy risks. The above experimental outcomes further confirm the reliability and practical applicability of the log anomaly detection architecture built on federated learning in this study.

Table 4. Computational cost.

Method	Training time (minutes)	Testing time (minutes)
LogGPT	167.2	7.87
CNN	101.4	2.61
DeepLog	76.8	4.56
LogTAD	234.1	1.02
LogReT	434.5	47.4
FLMLog	982.1	54.2

Table 4 summarizes the time overhead of all compared approaches, with all experimental results averaged over multiple datasets. Consistent with expectations, the proposed FLMLog method achieves optimal detection performance but requires greater computational overhead due to its abundant parameter scale. Even so, its inference efficiency is still competitive and tolerable compared with other LLM-based log detection frameworks, including LogReT.

6 Conclusion

In this paper, we propose a unified online log anomaly detection framework, FLMLog, which is based on federated Learning and large language model. We build a federated learning framework for distributed log anomaly detection. It only exchanges model parameters across participants to avoid privacy risks of centralized training. We test the framework on five public industrial log datasets, and experiments show that FLMLog achieves superior comprehensive performance, outperforming state-of-the-art methods in F_1-score on four of five datasets with notable improvements. FLMLog also shows strong adaptability to diverse log patterns, from simple HDFS to complex ThunderBird datasets, and maintains high precision and recall for effective anomaly identification.

References

1. A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, “Large-scale cluster management at Google with Borg,” in Proc. 10th Eur. Conf. Comput. Syst., 2015, pp. 1-17.
2. Junjielong Xu, Ruichun Yang, Yintong Huo, Chengyu Zhang, and Pinjia He. 2024. DivLog: Log Parsing with Prompt Enhanced In-Context Learning. In Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. 1-12.
3. Zhang, Chunkai , et al. "LayerLog: Log sequence anomaly detection based on hierarchical semantics." Applied Soft Computing (2023).
4. Nguyen, MT., Dinh, TD.L., Cao, V.L. (2025). Log-Based Representation Transferable Learning for Cross-System Anomaly Detection. In: Buntine, W., Fjeld, M., Tran, T., Tran, MT., Huynh Thi Thanh, B., Miyoshi, T. (eds) Information and Communication Technology. SOICT 2024. Communications in Computer and Information Science, vol 2351.
5. Zhou, Junwei , et al. "DRLLog: Deep Reinforcement Learning for Online Log Anomaly Detection." IEEE Transactions on Network and Service Management (2025):1-1.
6. Zhao, Nengwen , et al. "Identifying bad software changes via multimodal anomaly detection for online service systems." Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (2021).
7. Tao, Meiling , et al. "RoleCraft-GLM: Advancing Personalized Role-Playing in Large Language Models." (2023).
8. Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 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. 1285-1298.
9. H. Guo et al., “LogFormer: A pre-train and tuning pipeline for log anomaly detection,” in Proc. AAAI Conf. Artif. Intell., 2024, vol. 38, pp. 135-143.

10. Li, Jiangming , et al. "LogGraph: Log Event Graph Learning Aided Robust Fine-Grained Anomaly Diagnosis." Dependable and Secure Computing, IEEE Trans. on (T-DSC) 21.4(2024):14.
11. Siyu Yu, Pinjia He, Ningjiang Chen, and Yifan Wu. 2023. Brain: Log parsing with bidirectional parallel tree. IEEE Transactions on Services Computing 16, 5 (2023),3224-3237.
12. Xiao Han and Shuhan Yuan. 2021. Unsupervised cross-system log anomaly detection via domain adaptation. In Proceedings of the 30th ACM international conference on information & knowledge management. 3068-3072.
13. Van-Hoang Le and Hongyu Zhang. 2023. Log Parsing with Prompt-based Few shot Learning. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). 2438-2449.
14. S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, “SCAF-FOLD: Stochastic controlled averaging for federated learning,” in Proc. 37th Int. Conf. Mach. Learn., 2020, pp. 5132-5143.
15. Zeyang Ma, An Ran Chen, Dong Jae Kim, Tse-Hsun Chen, and Shaowei Wang. 2024. Llmparser: An exploratory study on using large language models for log parsing. In Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. 1-13.
16. Liu, Zihan , and N. Liao . "LogFA: Cross-system log anomaly detection based on hierarchical feature alignment and domain adversarial learning." 2025 2nd International Conference on Electronic Engineering and Information Systems (EEISS).
17. Aoxiao Zhong, Dengyao Mo, Guiyang Liu, Jinbu Liu, Qingda Lu, Qi Zhou, Jiesheng Wu, Quanzheng Li, and Qingsong Wen. 2024. Logparser-llm: Advancing efficient log parsing with large language models. In Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 4559-4570.
18. Y. Liu et al., “Symbiosis rather than aggregation: Towards generalized federated learning via model symbiosis,” IEEE Internet Things J., vol. 12, no. 20, pp. 41266-41275, Oct. 2025.
19. Hao Chen, Ruizhi Xiao, and Shuyuan Jin. 2021. Unsupervised Anomaly Detection Based on System Logs.. In SEKE. 92-97.
20. Y. Guo, Y. Wu, Y. Zhu, B. Yang, and C. Han, “Anomaly detection using distributed log data: A lightweight federated learning approach,” in Proc. Int. Joint Conf. Neural Netw., 2021, pp. 1-8.
21. Jieming Zhu, Shilin He, Pinjia He, Jinyang Liu, and Michael R Lyu. 2023. Loghub: A large collection of system log datasets for ai-driven log analytics. In 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE). IEEE,355-366.
22. X. Han, S. Yuan and M. Trabelsi, "LogGPT: Log Anomaly Detection via GPT," 2023 IEEE International Conference on Big Data (BigData), Sorrento, Italy, 2023, pp. 1117-1122.
23. H. Guo et al., “LogLG: Weakly supervised log anomaly detection via log-event graph construction,” in Proc. 28th Int. Conf. Database Syst. Adv. Appl., 2023, pp. 490-501.