Literature Review on Logs Management Dashboard and Real-Time Server Logs Processing

Varad Pundlik[1], Anirudha Udgirkar[2], Parth Tagalpallewar3, Divyank Sagvekar[4], Snehal Shintre[5]

[1]Student, SCTR's Pune Institute of Computer Technology, Computer Dept., Pune, Maharashtra, India, varadpundlik@gmail.com
[2] Student, SCTR's Pune Institute of Computer Technology, Computer Dept., Pune, Maharashtra, India, udgirkaranirudha@gmail.com
[3] Student, SCTR's Pune Institute of Computer Technology, Computer Dept. Pune, Maharashtra, India, parthtagalpallewar123@gmail.com
[4] Student, SCTR's Pune Institute of Computer Technology, Computer Dept. Pune, Maharashtra, India, divyanksagvekar37@gmail.com
[5] Assistant Professor, SCTR's Pune Institute of Computer Technology, Computer Dept. Pune, Maharashtra, India, spshintre@pict.edu

Abstract

Managing and analyzing server/application logs is a complicated process yet it is crucial for debugging and application development. Traditional logs monitoring methods make use of native scripting or softwares on servers like shell scripting. This paper provides an in-depth literature review on various log collection and analysis systems. As the complexity and scale of enterprise systems increase, the management of logs becomes a critical task for ensuring reliability, performance optimization, and system security. We analyze several architectures based on ELK, Splunk, Kubernetes, and cloud-native technologies, each tailored for different operational contexts such as business processes, networking, and containerized applications. Also development of log analysis techniques like root cause analysis based on logs and log classification are also reviewed in this paper. By examining existing research, we identify strengths, limitations, and future challenges in this domain, particularly in terms of anomaly detection, scalability, and real-time processing capabilities.

Keywords: Log analysis, Distributed log collection, ELK stack, Splunk, Real-time monitoring, Log storage management

1. Introduction

Logs play a crucial role in modern enterprise systems, logs help developers to fix errors in applications and find root cause analysis of an error [5]. Also it helps in analysis of application usage along with performance and non functional analysis of the system [6]. Logs save all the communication and procedure calls and network status in nodes so they are crucial in managing distributed system based applications [7]. When an issue arises, logs allow developers, system administrators, and support teams to trace the sequence of events, pinpoint where anomalies occurred, and identify potential causes. This reduces the time needed for manual intervention and helps quickly address system failures [8]. When an organization is using an application, they have to share the logs to partner or client organizations in case of testing as an evidence. Logs also serve as essential evidence that can be shared with clients, particularly when system performance issues or failures affect business-critical operations.

Logs can demonstrate the exact nature of the problem, when it occurred, and what actions were taken to resolve it. This level of transparency helps in establishing trust with clients, as it provides them with verifiable data regarding their systems. Furthermore, logs allow clients to have a comprehensive understanding of system events, enabling them

to audit or verify specific actions within their application ecosystems. Whether it's ensuring regulatory compliance, addressing security breaches, or verifying performance metrics, logs provide an authoritative source of truth. The vast amount of data produced by contemporary systems makes manual log analysis unfeasible. To make this process more efficient, automated frameworks for log collection and analysis have been developed, including cloud-native logging systems [3], Splunk [2], and the ELK stack [1]. With capabilities like pattern identification, anomaly detection, and configurable warnings, these technologies let businesses collect, store, and analyze logs in real time. Additionally, they enable the consolidation of logs from dispersed environments, providing a comprehensive picture of system performance and facilitating quicker problem-solving.

2. Literature review

2.1 ELK-based Distributed Log Collection

Work of the Literature [1] of the ELK architecture provides a strong foundation for understanding how distributed log collection can be achieved using open-source technologies. By integrating Apache Flume and Kafka, Zong presents a scalable solution for managing log data across multiple servers, offering a powerful way to standardize and collect logs from distributed sources. The implementation of Elasticsearch as the backbone of this architecture ensures fast querying and visualization of logs via Kibana. The log collection is achieved by creating a listener class functional module to bring uniformity in logs across various servers. And then Flume establishes the connection and sends logs with HTTP POST request into the redis and kafka message queue.Then the logs are stored in ElasticSearch Database and a Log Application module is created to show the logs on the dashboard. The trials' findings show that this approach is simpler and more effective at creating data than previous log analysis solutions. Flume is also capable of collecting the log data provided by the listening agent class. This system has the ability to gather, examine, and preserve important log data.more effectively and in accordance with a comprehensive study.

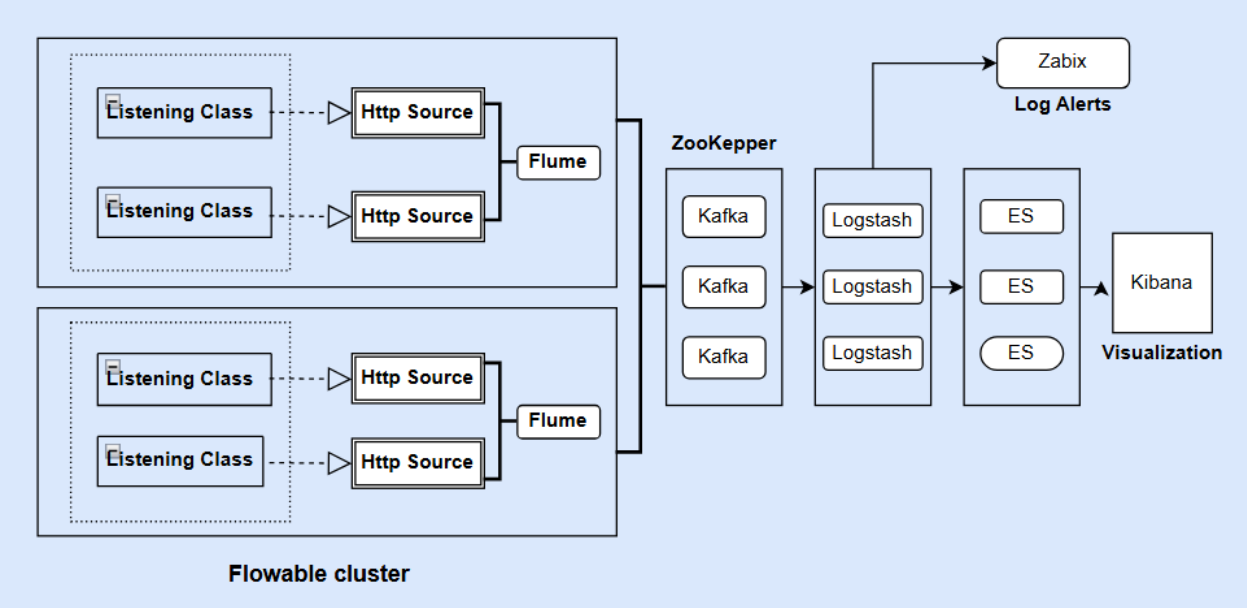

Fig. 1. Log Management System Architecture using ELK stack

The diagram represents a log processing pipeline where a **Listening Class** monitors inputs, an **HTTP Source** collects data, and **ZooKeeper** manages coordination. Logs are then sent to **Kafka** for message streaming, processed by **Logstash**, and stored in **Elasticsearch**. **alerts** are generated based on predefined conditions.

2.2 Splunk-based Log Collection

The literature [2] explains construction of a centralized logs collection system using splunk tool for FUTURE5 an university networking system.The logs were collected using network syslog and splunk forwarder and sent to logs management server.Numerous logs may be gathered by the log collection system, and these logs could be utilized for data analytics in addition to user statistics in order to examine user trends and offer services that are tailored to the needs of the users.

In FUTURE5, Splunk gathers system log files from about 440 hosts. Generally speaking, the collection from each host is automated, so there's no need to focus on the specifics of any given system output log when viewing the collected logs. A cross-sectional indexed log searchable is produced by compiling all of the system's raw access logs. Although the search function has been enhanced, it is challenging to keep searchable logs for an extended length of time due to capacity limits in the search system storage repository. This work's future scope was analysis of logs.

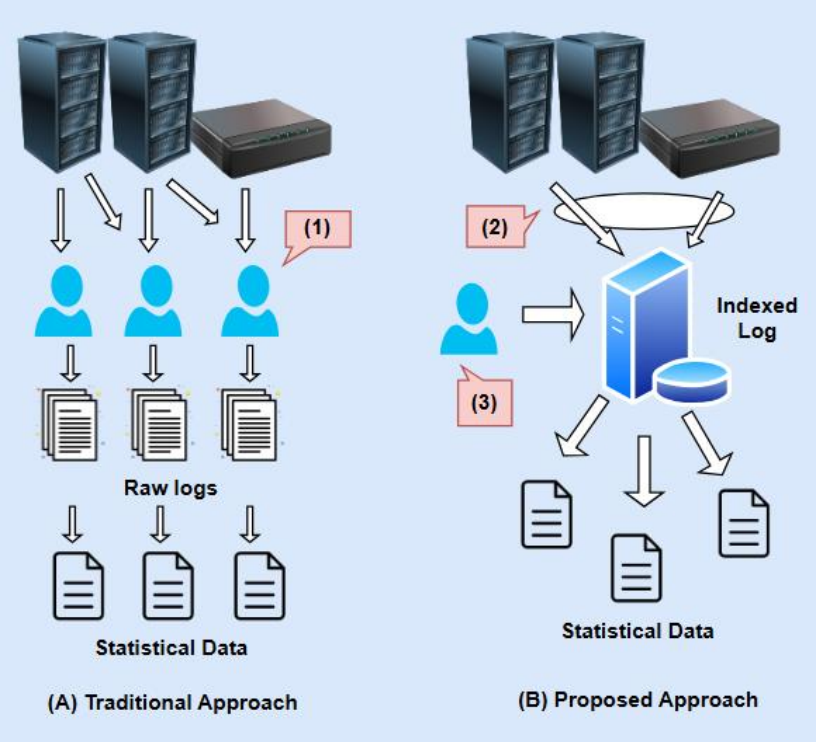

Fig. 2. Network log management using splunk

The diagram shows the **Traditional Approach** with slow raw log analysis and the **Modern Approach** with fast, indexed logs for better data use. This makes it quicker and easier for users to get insights. Indexed logs also improve data organization, and statistical data is generated more efficiently.

2.3 Cloud-Native Log Collection in Kubernetes

Using the Kubernetes cluster log storage architecture as an initial foundation, this study [3] investigates and creates a centralized platform architecture built around cloud technology that is crucial for enhancing the effectiveness of enterprise R&D. This literature develops a cloud based system to manage logs in the kubernetes cluster to troubleshoot pods related issues. A separate back-end is necessary for the distributed log architecture in order to store, examine, and query logs. Writing standard output (stdout) and standard error (stderr) streams is the most straightforward logging

technique for containerized programs. By using logrotate to evenly compress and dump the log of every container on the node, it is possible to successfully stop the server's disk from collapsing owing to an excess of log data.

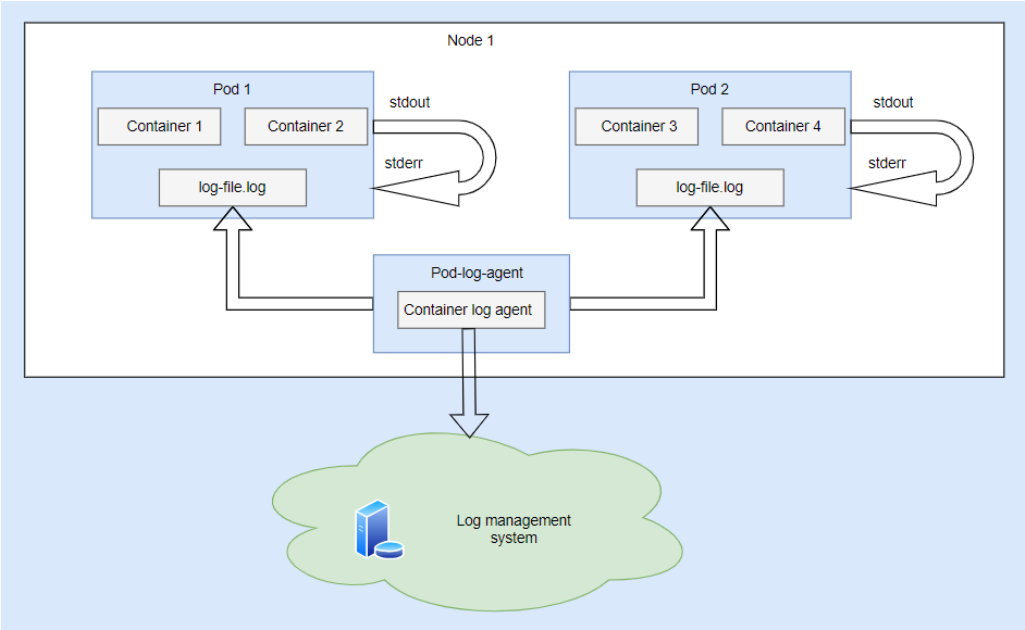

Fig. 3. Cloud based Log Management System using kubernetes.

The figure illustrates a logging setup in a containerized environment. **Pod** contains multiple containers that generate logs, which are collected using **stdout** and **stdin** methods from the pods. A **Pod Log Agent** and **Container Log Agent** manage the collection and forwarding of these logs. The logs are then sent to a **Log Management System** for processing and storage. This system ensures efficient handling of logs and enables centralized monitoring for better log management.

2.4 Real-Time System Log Monitoring

Gunasekaran et al. [4] propose a real-time log monitoring and analytics framework for the Jaguar XT5 supercomputer, which aggregates logs from multiple sources like RAS, net watch, and console streams. The framework processes these logs in real time to create materialized views, allowing administrators to promptly detect anomalies and profile applications. This real-time capability is essential for maintaining system reliability and optimizing resource usage while minimizing computational overhead—an especially important consideration for extensive systems. To detect anomalies, framework provides detailed insights into application behavior, helping administrators identify performance bottlenecks and improve system efficiency. However, challenges such as high computational demands and the difficulty of detecting anomalies during their first occurrence, especially with unknown events, persist. To resolve this constraints, future research could integrate machine learning and predictive analytics to enhance accuracy and speed of anomaly detection, potentially extending the framework's application to other supercomputing systems beyond.Jaguar.XT5.

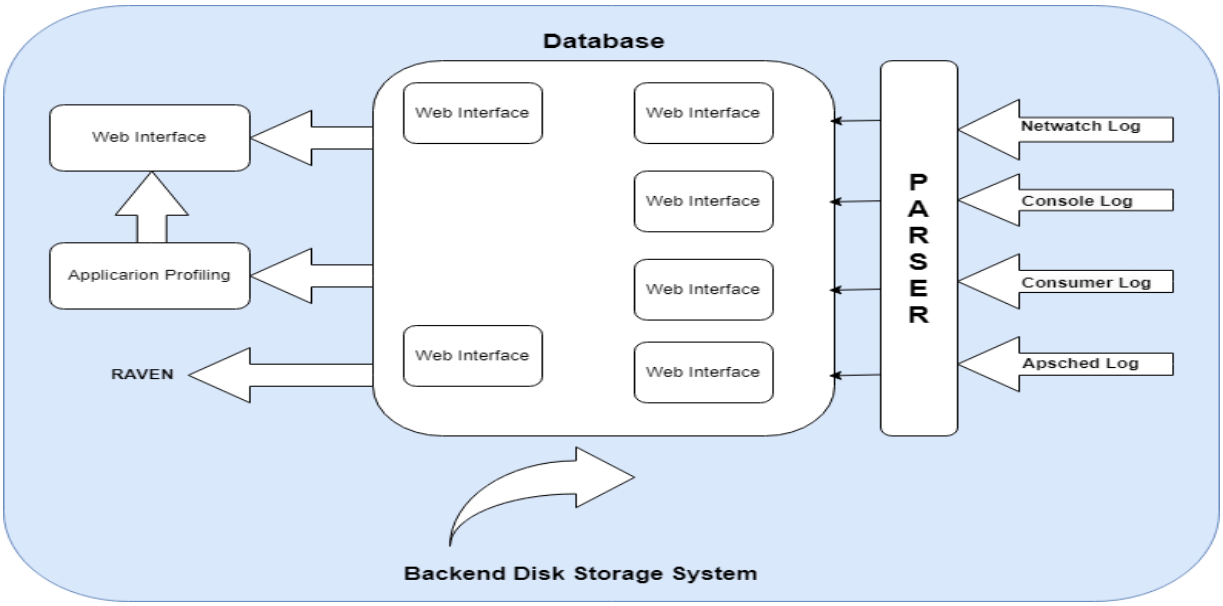

Fig. 4. Real time log monitoring framework

The figure shows a system where a **Web Interface** lets users interact with the application. The **Web Framework** helps publish the application and works with **Raven** to manage data. Different logs like **Networking Log**, **Console Logs**, **Consumer Log**, and **Apache Log** are collected and processed. These logs are stored in a **Database** and saved in a **Backend Disk Storage System** for easy access and management.

2.5 Automated Log Classification Using Deep Learning

Ramachandran's [6] deep learning framework offers a fresh take on automated log classification and anomaly detection, built to handle unstructured logs with impressive efficiency. Using a unique vectorization approach, "LogWord2Vec," combined with data augmentation, the model leverages a hybrid CNN-LSTM architecture to predict and classify errors. What sets this approach apart from traditional methods is its efficiency with smaller datasets, removing the dependency on massive datasets while still maintaining high accuracy. The blend of CNN's pattern recognition and LSTM's sequential memory allows the model to achieve a stellar accuracy rate of 99.19%. This breakthrough drastically reduces the manual workload associated with log classification, which is especially valuable in cloud or cluster-based environments where logs can accumulate at an overwhelming rate. Future research could expand this approach to tackle more complex, imbalanced datasets in diverse distributed environments.

2.6 Monitoring Access Log Database of information system on a Database Server

In the realm of distributed log collection and monitoring, Setiyadi and Setiawan [5] introduce a system specifically designed to monitor access to a MySQL database server, with a focus on enhancing data security through real-time user access tracking. The system employs a web-based interface constructed using PHP and MySQL, enabling administrators to closely monitor and control access to the database, particularly focusing on privileged users. Unlike general purpose log monitoring frameworks like the ELK stack which aggregates and processes logs from multiple servers this system narrows its focus on database security at the individual server level. It records detailed access logs for each user, allowing for more granular control over database access, and offers real-time insights that help prevent unauthorized access and mitigate the risk of privilege abuse. While the ELK stack emphasizes broad log aggregation across distributed systems, this approach stands out by prioritizing the security and integrity of database environments. By concentrating on access control, the system provides administrators with a critical tool to ensure that sensitive data

within the database remains secure. The study also underscores the increasing importance of comprehensive monitoring in database environments, particularly as the risk of unauthorized access and the abuse of privileges continues to grow. This work complements broader log management frameworks by offering a focused solution tailored to database access security, further emphasizing the necessity of monitoring privileged user activity to safeguard against potential security breaches

2.7 A Survey on automating Log Analysis for Reliability Engineering

Automated log analysis plays [7] a crucial role in managing the huge number of logs produced by modern software systems. Key areas of research include logging practices, compression of logs, parsing of logs, and mining of logs, with ML techniques like clustering and deep learning supporting methods such as detection of anomalies and prediction of failures. Tools like Splunk and Elasticsearch help automate these processes, though challenges such as scalability and real-time detection remain.

Log compression is essential for managing the huge volumes of logs produced by software systems, sometimes reaching rates of 50 GB per hour. To tackle the storage and analysis challenges posed by such large datasets, various compression techniques have been developed. These include bucket-based, dictionary-based, and statistics-based methods, each designed to reduce redundancy and improve storage efficiency. Bucket-based methods divide logs into blocks for parallel processing, optimizing efficiency by addressing both local and global redundancy. Dictionary-based approaches shrink file sizes by replacing repetitive elements like IP addresses with shorter references, while statistics-based techniques employ advanced models to identify patterns for dynamic compression. Collectively, these methods balance reducing storage demands with maintaining log data integrity for effective analysis.

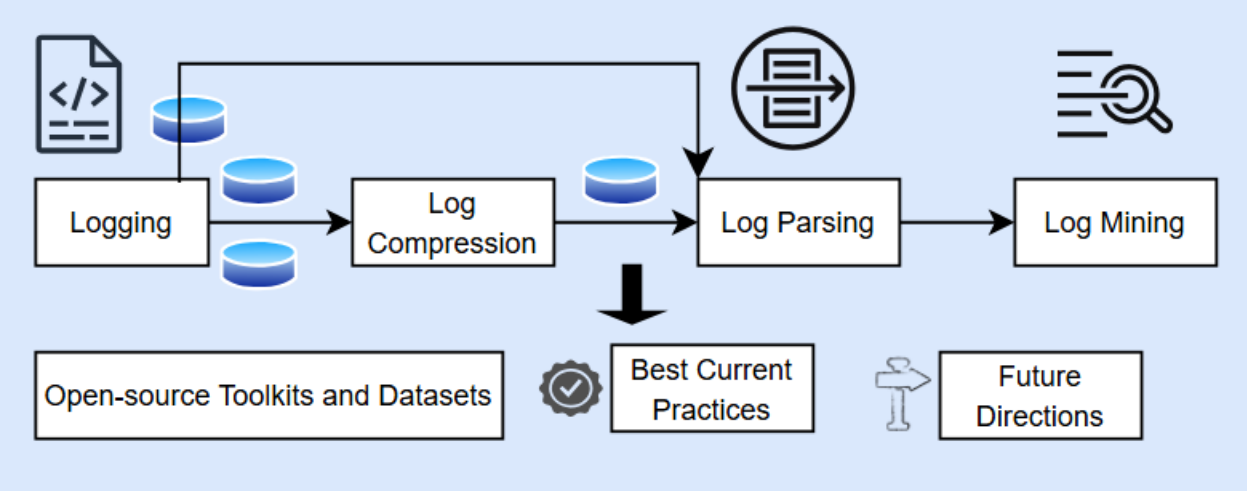

Fig. 5. Automated Log analysis process flow

The diagram outlines a log processing workflow, starting with **Logging** to capture data, followed by **Log Compression** to reduce size. The logs are then **Parsed** for easier analysis and **Min(ed)** to extract valuable insights. It highlights the use of **Open-source Toolkits** and **Datasets** for efficient processing. The system follows **Best Current Practices** and explores **Future Directions** to improve log management and analysis techniques.

2.8 Automatic RCA using LLM for Cloud Incidents

RCA plays a pivotal role in identifying, diagnosing, and resolving incidents in complex systems [8], particularly in cloud-based architectures, where the scale and intricacy of operations can lead to highly interdependent issues. Logs are a vital source of information in RCA, capturing detailed records of system events, errors, and performance anomalies. However, relying solely on logs often overlooks key insights that can be gained from additional sources such as system metrics, telemetry data, and traces, which together provide a comprehensive picture of system behavior and the interactions across components. To effectively perform RCA, the process begins with the collection of multi-source data from logs, parameters of system performance (e.g. memory usage), and distributed traces that show how requests propagate through various services.

Once this data is collected, data analysis techniques such as parsing of logs,detection of anomalies and correlation analysis can be applied. These techniques allow analysts to identify patterns, anomalies, or relationships within the data that could be indicative of a potential root cause. After analyzing the data, hypotheses about the root cause are formulated based on observed patterns and verified through further testing and analysis. To streamline this process and enhance accuracy, machine learning models can be employed to automate the prediction of incident root causes. These models learn from historical data to predict and categorize new incidents while providing explanations for their predictions. By integrating automated RCA workflows and ensuring the system continually adapts to new types of incidents, organizations can reduce the manual effort required in root cause identification and improve their overall system reliability and responsiveness

3. Findings in Literature Survey

The literature survey provides a comprehensive overview of various log management and analysis systems, each contributing insights relevant to the implementation of the centralized log management platform described in the abstract. The review reveals that modern enterprise systems produce an enormous volume of logs, making manual analysis infeasible. Several log management architectures, such as ELK, Splunk, and Kubernetes-based systems, are discussed, emphasizing their strengths in scalability, anomaly detection, real-time processing, and log storage.

Table 1. Findings in Literature Review

Sr. No	Author	Research method	Methodology	Findings
1.	Y. Zong et al. [1]	Log collection using ELK architecture	This literature focuses on creating a distributed log collection system using ELK architecture. It uses flume and kafka along with ELK stack to collect log files.Flume establishes the connection and sends logs with HTTP POST request into the redis and kafka message queue.Then the logs are stored in ElasticSearch Database and a Log Application module is created to show the logs on the dashboard.	ELK stack (Elastic) is an industry popular tech stack used for data engineering and especially for logs management. This literature explains the construction of a distributed log collection system using ELK stack which increases the scalability of the system and can handle large volumes of log data.

2.	Masaru Okumura et al. [2]	Log collection using Splunk	In this paper a splunk based log collection system is developed to collect logs of FUTURE5 university networking system using network syslog and splunk forwarder.	Splunk is an alternative tool to ELK stack for Data Engineering and Logs Management.The network logs collected in this literature can be utilized for system monitoring as well as data analytics operations.
3.	T. Chen et al. [3]	Log Collection using Cloud Technology	In this literature a cloud based log collection system is built using kubernetes cluster.It collects standard output and standard error streams of the system as logs.	Since kubernetes cluster and are used for logs storage the cloud based system is distributed and the system is scalable and flexible.
4.	Raghul Gunasekaran et al. [4]	Real-Time System Log Monitoring Analytics Framework	This paper proposes a real time log collection system for Jaguar XT5 supercomputer collecting network,console, consumer and apsched logs in a MySQL database and Dashboard is created using Apache Web Server	This system enables anomaly detection and real time processing,it reduces system downtime and optimizes resource usage.
5.	Shekar Ramachandran et al [5]	Automated Log Classification Using Deep Learning	This literature focuses on classifying unstructured logs, using a unique vectorization approach, "LogWord2Vec," combined with data augmentation, the model leverages a hybrid CNN-LSTM architecture to predict and classify errors.	This paper supports the abstract's goal of using Large Language Models (LLMs) for root cause analysis, classification of logs in process wise or API request wise on the logs management dashboard and machine learning analytics for performance monitoring
6.	A Setiyadi et al. [6]	Monitoring Access Logs of Information Systems on a Database Server	This literature proposes a web-based system using PHP and MySQL to monitor MySQL database server access. The system tracks user activity in real-time, focusing on privileged users. Unlike general log monitoring frameworks, it emphasizes database security at the individual server level, offering detailed access logs for granular control.	The study highlights the system's capability to enhance database security through real-time monitoring of user access. By focusing on privileged user activity, it helps prevent unauthorized access and privilege abuse. This approach complements broader log

				management frameworks, emphasizing the critical need for database-specific monitoring to mitigate security risks effectively.
7.	Shilin He et al. [7]	Log Analysis Automation	Automated log analysis is vital for managing the vast volumes of logs generated by modern software systems [7]. Research in this field explores areas such as logging techniques, log compression, parsing, and mining. Machine learning methods, including clustering and deep learning, are used to support tasks like anomaly detection and failure prediction.	Log compression methods optimize storage and analysis of large datasets, managing up to 50 GB per hour. Bucket-based techniques enhance efficiency through parallel processing, dictionary-based methods reduce redundancy via symbolic replacements, and statistics-based approaches dynamically compress logs by identifying patterns, striking a balance between storage efficiency and data integrity.
8.	Yinfang Chen, Huaibing Xie[8]	Root Cause Analysis using LLM	This study focuses on automating Root Cause Analysis (RCA) by collecting multi-source data from logs, system metrics, and distributed traces. Data analysis techniques, including log parsing and anomaly detection, are applied, followed by the use of ML models to predict and explain root causes.	The integration of machine learning models in RCA workflows enhances accuracy and efficiency by predicting root causes from historical data. This approach reduces manual effort and improves system reliability and incident response
9.	S. Chabridon [9]	Improving Performances of Log Mining	statistics is collected from diverse resources like syslog and application logs, then parsed into dependent information for analysis. Anomaly detection, through system studying or rule-based total techniques, identifies uncommon patterns in logs that may suggest problems.	Key findings from a log control challenge include the identification of ordinary problems, highlighting regions for improvement or automation. false alarms and misclassifications screen where log messages don't as it should be represent the gadget's state.

10.	Anton Chuvakin et al [10]	Logging and Log Management: The Guide to Logging.	Facts collection and normalization, gathering logs from various sources and making sure they observe a constant layout. This addresses the common challenge of handling disparate log formats. subsequent, recall covert logging techniques to create a comfortable and stealthy monitoring setup that detects threats discreetly, improving typical security without alerting capacity attackers.	powerful log control calls for more than simple facts aggregation. by normalizing and covertly amassing information, organizations can use logs to hit upon and respond to security incidents. Correlating diverse logs and studying them with statistical strategies enhances the capacity to foresee and mitigate dangers, providing companies a comprehensive protection towards threats.
11.	Zhang et al [11]	Log Parsing Using Prompt	The method entails designing a log parser that relies on semantic sample reputation in preference to static token matching. Through using a few-shot getting to know, a small sample of categorized logs is analyzed to train models that identify commonplace log structures (templates) and variables (parameters) across numerous logs.	Logs provide critical statistics for troubleshooting and maintaining software program systems, especially for tasks like anomaly detection, root purpose evaluation, failure prediction, and log compression.
12.	Lyu, M. R. et al [12]	Tools and benchmarks for automated log parsi	Logs are important for diagnosing and coping with software program structures but are hard to analyze due to excessive extent and unstructured records. conventional log parsing strategies, consisting of regex, are manual and error-inclined. more modern, records-pushed parsing strategies like sample mining and clustering permit automatic template extraction, making evaluation extra efficient. Open-source tools and benchmarks have further driven industry adoption.	This assignment entails gathering diverse log facts, the usage of computerized log parsers (e.g., SLCT, Spell, Drain) to structure it, and comparing these tools primarily based on accuracy, robustness, and efficiency. An open-supply toolkit could be integrated and benchmarked to guide choice. The manner may even report sensible insights for a hit log management deployment.

13.	Pereira, G. A [13]	Log-based software monitoring	The technique of this take a look at entails a systematic mapping method to investigate research in log-based totally software program monitoring. First of all, initial searches throughout five facts resources were conducted to define seek criteria and compile a listing of probably relevant research. Inclusion and exclusion standards had been carried out to refine this list to key research as much as 2018.	The findings advise that logs are essential for diagnosing gadget problems, but their high extent and unstructured nature make manual analysis impractical. automatic log parsing techniques, along with statistics-pushed strategies like sample mining and clustering, provide huge enhancements in structuring logs for evaluation.
14.	Pianese, F[14]	System Log Parsing	This paper surveys log parsing solutions by providing a comprehensive taxonomy and empirically analyzing the performance of 17 open-source parsers. The evaluation is both quantitative and qualitative, highlighting key features and alternative approaches.	By evaluating various log parsers, the study highlights their strengths and weaknesses, offering guidance on selecting open-source solutions. It also discusses future challenges and research opportunities in log parsing for system management.
15.	M. R. Lyu [15]	Log Parsing Use in Log Mining	This study evaluates toolkit and testing their performance across five datasets with over ten million log entries, examining effects on downstream mining tasks .	The study identifies six important insights on parser efficiency and accuracy, providing valuable guidance for developers in selecting optimal parsers for large-scale, automated log mining in complex systems.
16.	Gang Wu [16]	Log Punctuations Signature for Log Parsing	This study introduces LogPunk, an innovative online log parsing approach. It uses a signature method based on punctuations of the logs and length of message.	LogPunk demonstrates high efficiency and robustness, outperforming five log parsers with a 91.9% accuracy on 16

			and avoid traditional tree structures, LogPunk enables efficient template matching through candidate sets and similarity functions, ensuring rapid and accurate parsing	datasets from LogHub, proving effective for large-scale log parsing in complex systems.

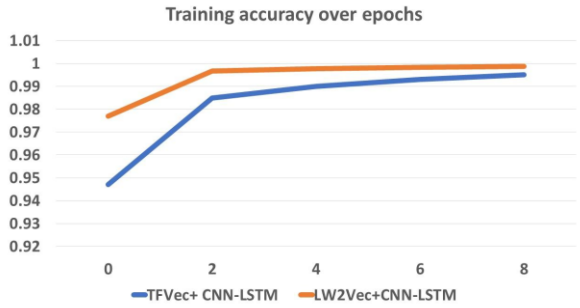

Fig. 6. Training accuracy vs. no. of epochs in Log classification using Deep Learning

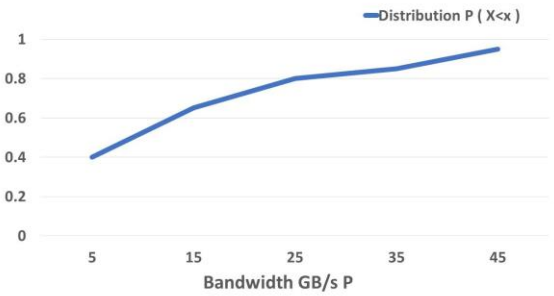

Fig. 7. CDF vs. Bandwidth distribution in Real time log monitoring system

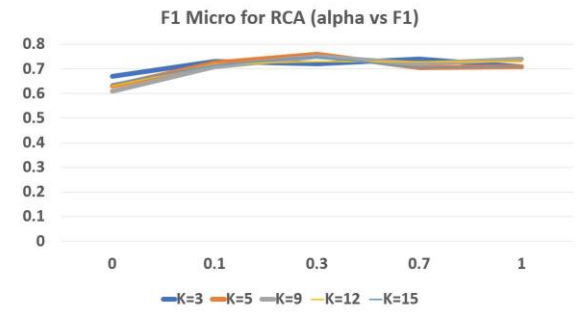

Fig 8. F1 Micro score vs. alpha for various k values in Automated Logs RCA

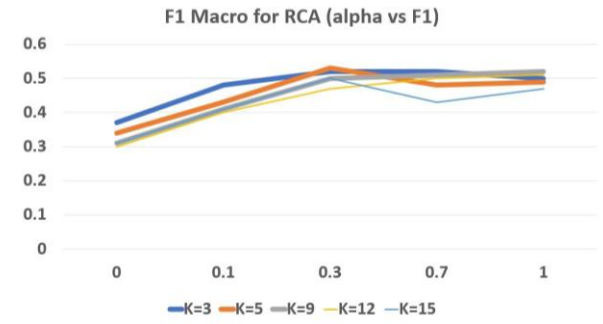

Fig. 9. F1 Macro score vs. alpha for various k values in Automated Logs RCA

4.Algorithms used in this work

The Dynamic Log Compression Algorithm (DLCA) enhances log analysis and storage by compressing repetitive sequences through pattern recognition and adaptive techniques. It optimizes timestamps and reduces storage needs while enabling faster log retrieval.While developing a log collection system it can be used for compressing older logs to reduce disk usage.

1. Start
2. Group log entries by user into sequences.
3. For each user sequence, identify matching patterns of actions.
4. If a pattern matches:
    a. Compress the sequence into a token with its start and end timestamps.

   b. Clear the sequence after compression.
5. If no pattern matches, append the log entry as-is to the compressed output.
6. Repeat for all log entries and patterns.
7. End

Significance:

Dynamic log compression algorithm is important in log management because it compresses redundant data through pattern recognition and adaptive techniques.The reviewed literature emphasizes the necessity of log compression to handle the vast volume of logs generated by modern enterprise systems. Automated log analysis techniques, such as those using machine learning and deep learning, enhance system performance by reducing storage overhead and improving log retrieval speed [7].

5. Performance Evaluation Parameters

Log collection system evaluation parameters:

1. Throughput Rate: The number of data records processed or written per second.
2. Total Data Written: The total number of data records processed during the test.
3. Data Transfer Rate: The volume of data written to the system per second, measured in megabytes (MB/s).
4. Write Latency: The average time taken to write a single piece of data, measured in milliseconds (ms).
5. System Scalability: The system's ability to maintain stable performance (e.g., collecting log data) under high concurrency and distributed execution.

Model and data training evaluation parameters:

1. Input Parsing Rate: Processes and groups log entries by users at a dynamic rate for further analysis.
2. Pattern Matching Efficiency: Identifies and matches action patterns in user sequences with high accuracy and speed.
3. Compression Tokenization Rate: Generates unique tokens for matched patterns while optimizing metadata storage.
4. Timestamp Optimization: Captures and stores start and end timestamps for compressed patterns to reduce redundancy.
5. Dynamic Adjustment Frequency: Dynamically clears processed sequences to ensure optimal system performance and maintain scalability.
6. Output Generation Rate: Produces compressed logs and unmatched entries in JSON format for efficient storage and retrieval.

Significance:

When there are different architectures in development of a log collection system, Performance Evaluation plays the pivotal role in choosing the correct log collection system as per use case and requirements.The reviewed works on Splunk-based log collection [2] and ELK stack implementations [1] demonstrate the need for high-throughput, low-latency log processing solutions in large-scale environments.

6. Conclusion

In conclusion, this literature review highlights the growing significance of effective log management systems in contemporary enterprise environments. As systems become more complex, the collection and analysis of logs are essential for maintaining reliability and performance.

Various architectures, including ELK, Splunk, and cloud-native Kubernetes systems, offer different strengths and limitations, providing organizations with options tailored to their needs. However, challenges such as scalability and real-time processing continue to exist across all approaches.

In ELK stack implementation With throughput rates exceeding 3,000 logs per second and write latencies averaging 48,500 ms in high-concurrency environments, performance evaluation underscores the importance of scalability and efficiency in these systems.

Future research should aim to enhance scalability, storage efficiency, and automation to meet these evolving demands. This may involve implementing more efficient log aggregation techniques, optimizing data storage methods, and utilizing containerization for flexible deployment in cloud environments.

Additionally, future developments should explore the integration of advanced technologies such as Large Language Models (LLMs) for automated root cause analysis with F1 Micro score 0.689 and F1 macro score 0.51 respectively, machine learning for performance monitoring where log classification using deep learning paper had 99.9% training accuracy., and full-text search databases for improved log extraction and analysis.

Automating log parsing, setting up alerts, and employing auto-healing scripts can further streamline operations, making log management systems more efficient and adaptive. As companies increasingly adopt cloud computing and distributed architectures, the need for robust, scalable, and intelligent log management solutions will grow, with resilience and adaptability being key focal points for future innovations in this critical area.

References

[1] Yize Zong, "Distributed log collection for business processes based on ELK architecture," IEEE 2nd International Conference on EEBDA, Changchun, China, 2023, pp. 1523-1529

[2] Masaru Okumura et al.. Constructing a Log Collecting System using Splunk and its Application for Service Support. 2016 ACM SIGUCCS Annual Conference (SIGUCCS '16). Association for Computing Machinery, New York, NY, USA, 103–106

[3] T. Chen et al., "Design of Log Collection Architecture Based on Cloud Native Technology," 2023 4th Information Communication Technologies Conference (ICTC), Nanjing, China, 2023, pp. 311-315

[4] Gunasekaran, Raghul et al. Real-Time System Log Monitoring/Analytics Framework. United States,2011.

[5] Shekar Ramachandran et al.,"Automated Log Classification Using Deep Learning", Procedia Computer Science, Volume 218, 2023, Pages 1722-1732, ISSN 1877-0509

[6] Setiyadi, Angga et al., Information System Monitoring Access Log Database on Database Server. IOP Conference Series: Materials Science and Engineering. 407. 012110. 10.1088/1757-899X/407/1/012110.

[7] Shilin He et al., "A Survey on Automated Log Analysis for Reliability Engineering". ACM Comput. Surv. 1, 1 (June 2021)

[8] Yinfang Chen et al, "Automatic Root Cause Analysis". Nineteenth European Conference Association for Computing Machinery., New York, NY, USA, 674–688.

[9] S. Chabridon et al, "Improving Performances of Log Mining," IEEE International Symposium on Modeling of Computer and Telecommunication Systems (MASCOTS)., Milwaukee, WI, USA, 2018, pp. 237-243
[10] Anton Chuvakin et al"Logging and Log Management ",2013, Pages 71-91, ISBN 9781597496353
[11] Zhang et al, "Log Parsing Using Prompt" . arXiv. https://arxiv.org/abs/2302.07435
[12] Lyu, M. R. (2019) et al. "Automated Log Parsing Tools". arXiv. https://arxiv.org/abs/1811.03509
[13] Pereira, G. A et al "A Log-based mapping study". arXiv. https://arxiv.org/abs/1912.05878
[14] Pianese, F et al. (2023). "Survey of Log Parsing in a System" . IEEE Transactions on Data Engineering, 1–20.
[15] M. R. Lyu et al, "Log Parsing Use in Log Mining", 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Toulouse, France, 2016, pp. 654-661
[16] Gang Wu et al. "Log Punctuations Signature for Log Parsing Applied Sciences 11, no. 24: 11974.