Log Frequency Analysis for Anomaly Detection in Cloud Environments at Ericsson

Prathyusha Bendapudi
Blekinge Institute of Technology
Karlskrona, Sweden
prbe21@student.bth.se

Vera Simon
Ericsson
Aachen, Germany
vera.simon@ericsson.com

Deepika Badampudi
Blekinge Institute of Technology
Karlskrona,
Sweden
deepika.badampud
i@bth.se

***Abstract*—Log analysis monitors system behavior, detects errors and anomalies, and predicts future trends in systems and applications. However, with the continuous evolution and growth of systems and applications, the amount of log data generated on a timely basis is increasing rapidly. This causes an increase in the manual effort invested in log analysis for error detection and root cause analysis. The current automated log analysis techniques mainly concentrate on the messages displayed by the logs as one of the main features. However, the timestamps of the logs are often ignored, which can be used to identify temporal patterns between the logs which can form a key aspect of log analysis in itself. In this paper, we share our experiences of combining log frequency based analysis with log message based analysis, which thereby helped in reducing the volume of logs which are sent for manual analysis for anomaly detection and root cause analysis.**

***Index Terms*—Log Analysis, Log Frequency Patterns, Anomaly Detection, Machine Learning, Cloud Environments**

I. INTRODUCTION

Software logs have been widely employed in a variety of reliability assurance tasks because they are some of the main sources of information available that record software run-time information. System logs are essential for various reasons, especially for monitoring and troubleshooting in a given computing environment [7]. However, the exponential growth of log data presents challenges for manual analysis, leading to the development of automated log analysis tools that enhance efficiency and effectiveness.

Traditional methods of detecting anomalies in logs involve using manual techniques such as searching for specific keywords (such as "failed", "exception", and "error") or match- ing rules to identify potentially problematic logs associated with system issues [2]. However, with the growing volume and variety of logs generated by modern software systems, these manual approaches are becoming increasingly impractical due to their labor-intensive nature [2].

Automated log analysis tools for different phases such as logging, log compression, log parsing, and log mining exist [5]. In the log mining phase, statistics, data mining, and machine learning techniques can be employed for automatically exploring and analyzing large volumes of log data to obtain meaningful patterns and informative trends. In existing log analysis techniques, the log analysis is primarily based on log messages [5]. Little consideration is given to the timestamps of the logs, which might also serve as an important source of information with regard to the temporal pattern displayed by a set of logs [1]. Few solutions focused on identifying temporal patterns using deep learning methods in log analysis [3, 1]. However, the time frequency patterns displayed by the logs were not correlated to the processes indicated by these logs and the messages displayed by the logs. Baril et al. developed a new model that captures temporal deviations by means of a sliding window data representation [1]. However, the model they developed used a semi-supervised learning approach, where new temporal patterns were compared to past temporal patterns, which were devised to be 'normal' and anomalies were predicted based on the deviation between old and new temporal patterns. Due to the continuous evolution of software processes and ever-changing patterns in log occurrences, an unsupervised learning approach could be beneficial for log-based anomaly detection. Ericsson developed an AI-assisted tool, Lexicon, for automated log based anomaly detection and root cause analy- sis in 5G Cloud network infrastructure. By employing ma- chine learning algorithms like clustering and TF-IDF (term frequency-inverse document frequency), Lexicon automates anomaly detection and root cause identification based on log messages (identifier-based approach). It operates in two modes: monitoring and troubleshooting, facilitating proactive issue detection, and retrospective analysis.

Lexicon takes logs generated from different processes in different cloud environments of Ericsson (Kubernetes pods creation, VM creations, health checks, application tests, etc.), and analyzes these logs primarily based on the log message. Lexicon also detects erroneous logs based on the messages displayed by the logs. It categorizes all the input logs into different groups (log types) based on the similarity of their message and the process that generated these logs. This categorization is done for both normal and erroneous logs. Lexicon assigns a 'normal value' to each log, which is on a scale of -1 to 10 based on how likely the log is to be a normal log without any anomaly or error. Here, an anomaly refers to any unexpected behaviour or error in the associated cloud environment. The details on the normal value in the context of Lexicon are provided in Table I. Lexicon sends the logs with very low normal values to system experts for

manual analysis to understand the root cause of the error that the log might represent. Although Lexicon reduces the manual effort involved in error detection and root cause analysis to a great extent, the combination of the timestamp-based analysis approach with Lexicon's existing identifier-based approach can prove to be beneficial in further reducing the manual effort involved in anomaly detection and root cause analysis.

The aim of this study is to streamline log analysis for root cause detection in Ericsson's cloud environment by reducing the system experts' effort involved in log analysis. This is achieved by combining the two approaches of log analysis: Identifier-based and Timestamp-based. Once Lexicon classifies the logs based on an identifier-based approach, we further classify them using a timestamp-based approach, which includes identifying frequency patterns in logs corresponding to different processes and flagging deviant logs as anomalies.

II. RELATED WORK

Previous research includes developing new approaches for automated log analysis, however, the existing ap- proaches consider only one approach of analysis, i.e., either identifier(message)-based or timestamp-based [6]. We wanted to see if we can improve the efficiency of log analysis-based anomaly detection and root cause analysis by combining both approaches. Moreover, the different log analysis tools available currently, such as splunk, logSayer [13], log assist [11], logFlash [9] provide automated log analysis solutions, but none of them explored the timestamp-based approach yet. Most automated log analysis tools concentrated on identifier-based methods, demonstrating a gap in explor- ing the timestamp-based aspect. Prior research, such as that by Zaman et al. [12], presented tools like SCMiner that localize system-level concurrency faults through log partitioning. While effective in detecting concurrent system failures, SCMiner primarily addresses known system failures and lacks consideration for detecting anomalies based on their time of occurrence. Similarly, LogAssist by Chen et al. [11] focus on summarizing logs to condense large log files into informative summaries through log partitioning. However, it does not consider the detection of anomalies based on their time frequency patterns. The proposed log analysis in this paper advances log partitioning by combining it with feature extraction, forming a good base for further anomaly detection. Grund et al. [4]'s approach, grounded in clustering and statistical methods, abstracted logs automatically. In comparison, this paper builds on these methods

III. STUDY DESIGN

In this section, we describe the steps followed to streamline log analysis for root cause detection in Ericsson's cloud environment. The steps conducted in this study are illustrated in Figure 1.

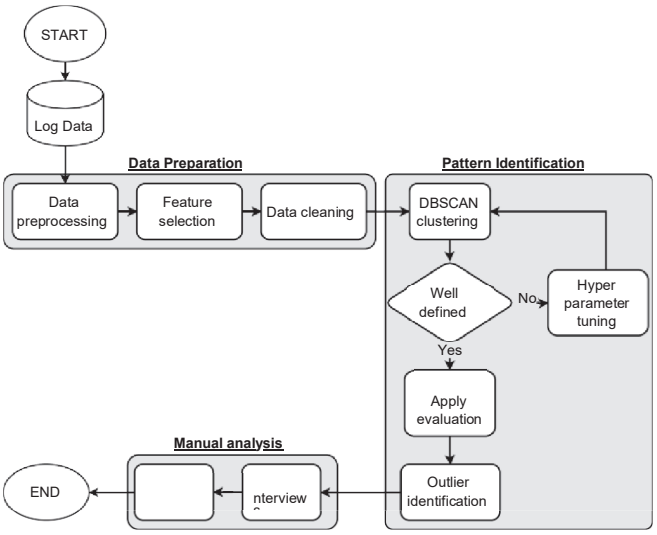


Fig. 1. Research workflow

A. Data Preparation

The initial dataset comprises seven JSON files, each encompassing a complete day's worth of system logs. The dataset contains an extensive collection of approximately 10 million logs which were initially analysed by Lexicon using the identifier based approach, offering a rich source of data for analysis. In each log, several fields were captured in a structured manner, including a unique log uuid, log type, log input type, timestamp, message, and cleaned message, among others. In order to prepare the initial dataset for comprehensive analysis, a series of data preprocessing steps were implemented. The seven JSON files, each representing a single day's log data, were combined into a single dataset to prepare the logs to be grouped on the basis of their log type. A Python script was written to extract and transform the structure of the log entries and store all these entries in a data frame. Each column within the data frame corresponds to a specific log field, such as log uuid, log type, message, timestamp, and many more.

Since it is computationally complicated to deal with all the fields of a huge number of logs, necessary fields were extracted before proceeding with the analysis. Log ID, log type, timestamp, normal value, and message were the required fields for the frequency analysis. Important fields from the log data were selected for the analysis based on what would support the frequency analysis and what log fields would be needed for the system experts to analyze the logs efficiently. A Python script was written to extract the necessary fields from the data set and store them in a separate data frame. While forming this data frame, the logs were grouped according to their log types. Approximately 2000 log files were generated, each of which contained logs pertaining to a particular process in the cloud environment.

The log data contained no null values or significant anoma-

lies that needed removal or replacement. For our analysis, we excluded some log files that contained fewer than 100 logs. This was because we had sufficient examples of log types with more data, which would lead to a more representative result for this kind of analysis. Also, one of the targets of this research is to reduce the amount of logs sent for manual analysis. It is more effective if the analysis is run on the log types that appear several hundreds of times.

The first author converted 7 JSON files, each representing logs from one day of the week, into approximately 2000 files. Each of these new files contained logs of one log type. Then the first author excluded files with less than 100 logs from the 2000 files.

The final dataset consists of the fields listed in table I.

TABLE I
LOG FIELDS

Log uuid	It is a unique identifier for each log.
timediff	It is the time difference between the current log and its preceding log in seconds.
log type	It is a hash ID that is unique to each log type, which depicts a particular process occurring in the cloud environment.
normal value	It is a value assigned to a particular log type by Lexicon according to the degree of normalcy shown by the log occurrences on a scale of -1 to 10, where 0 means that the log is abnormal/erroneous, and 10 means that the log is normal. A normal value of -1 indicates that the log hasn't been seen before and is new to Lexicon.
message	The raw message displayed by the log

B. Pattern Identification

The prepared data set consisted of log files, each log file consisting of logs corresponding to one log type, which represents a particular process in the cloud environment. It was necessary to identify the temporal patterns of these logs in order to proceed with the study. DBSCAN algorithm was used to identify the frequency patterns in each log file. Traditional clustering methods like K-means and KNN may struggle with identifying time-based patterns, especially when clusters occur at irregular intervals [10]. In these clustering algorithms, the number of clusters needs to be mentioned explicitly, and since there was no prior information on how many temporal patterns could be identified in one log file, it was not possible to use them for this analysis. DBSCAN, however, is useful in detecting clusters based on temporal proximity.

Each log file, representing a specific log type, was sequentially loaded for clustering. Parameters such as epsilon(ε) and min_samples were meticulously calibrated to optimize clustering results, with an iterative hyperparameter tuning process ensuring alignment with the temporal dynamics inherent in the log data. The importance of each hyperparameter in the context of log frequency analysis is as follows:

- **Epsilon (ε) -** Defining Neighbourhood Radius: The epsilon parameter served as a pivotal determinant in delineating the radius within which data points were considered neighbors. In the context of log frequency analysis, epsilon played a crucial role in capturing the temporal proximity of log occurrences. A range of epsilon values were explored with meticulous tuning to strike a balance between granularity and inclusivity.
- **Min_samples -** Determining Cluster Density: This parameter emerged as a vital factor in shaping the clustering process. It dictated the minimum number of data points required to constitute a cluster, a critical consideration given the varied frequencies of log occurrences. For example, some log files can have 100 logs but some other can have 500 logs. Also, in each log file, a number of logs can have one temporal pattern and some other logs can have a different temporal pattern. In log frequency analysis, fine-tuning min_samples is crucial to adapt the algorithm's sensitivity to density fluctuations displayed by different log types.
- **Custom Distance Metric -** Tailoring for Log Characteristics: The custom distance metric was intricately crafted to encapsulate the temporal dissimilarity between log occurrences, a fundamental aspect in log frequency analysis. The main challenge faced while implementing DBSCAN on the log data was that each log type had different time intervals between consecutive logs, and it was not possible to use the same set of hyperparameters for all the log types and achieve the same efficiency in clustering. A fixed inter-cluster distance could not be used for all the clusters in all the log files. For a cluster in which the time difference between consecutive logs is 1 second, a distance of 1 second will be huge. On the contrary, for clusters where the time difference is in minutes or hours, 1 second distance will be insignificant. Usage of this custom distance made it possible to use constant values for the hyperparameters in every cluster, especially for epsilon, which otherwise would have to be tuned for every log type. Using a custom distance metric enabled normalizing the variance within the clusters based on the time difference between consecutive logs.

The identified patterns were systematically analyzed, considering cluster statistics and visualizations such as scatter plots. We considered evaluation metrics to serve as objective criteria for selecting log files for in-depth analysis. Details about the evaluation metrics:

- Mean by Standard Deviation value: This metric is used to measure the dispersion among the data points in a cluster. A higher dispersion means that the clusters are not well defined, and the gap between data points is bigger. A low mean by standard deviation value indicates that the clusters are well-defined.
- Outlier by Datapoint count ratio: This metric is the ratio of the number of outliers in a log file to the total number of logs in the log file.
- Silhouette score: This metric measures how well a dataset is clustered by measuring intra-cluster similarity and inter-cluster discrimination.

Subsequently, outlier logs were extracted for further manual analysis, aimed at facilitating error detection within the log data.

C. Manual Analysis

The manual analysis phase commenced after applying the DBSCAN clustering algorithm and identifying outliers us- ing predefined evaluation metrics. These metrics guided the selection of files that could be sent for detailed analysis. The manual examination aimed to reveal the relation between specific log messages and temporal patterns shown by them, providing insights into error detection and root cause analysis implications. All the logs that deviated from the identified frequency patterns were termed outlier logs. The legitimacy of the outliers was supported by using evaluation metrics for the DBSCAN clustering algorithm. Some of the outlier logs were taken and manually analyzed to identify the cause of the deviation. The outlier logs were then sent to system experts to learn more about these logs and to identify if these outliers indicated anomalies in the processes.

Semi-structured interviews [8] were conducted with system experts to extract insights into manual log analysis. Practitioners from Ericsson were selected for the interviews based on expertise in log analysis and domain knowledge. The interviews aimed to understand the relevance of log frequency analysis for reducing manual effort and to explore connections between log processes and temporal pat- terns. A collaborative effort with manual log analysts followed, involving a curated selection of log files shared for analysis. Analysts were briefed on the current functioning of Lexicon and the proposed improvements to log analysis before conduct- ing a thorough manual analysis. Subsequent semi-structured interviews enabled analysts to share their observations and insights, focusing on potential errors indicated by logs, outlier significance, and the approach's feasibility of reducing man- ual workload. Interview topics ranged from the professional backgrounds of the practitioners to the methodologies used for log analysis and encountered challenges. The aim of the interviews was to understand the impact of the proposed approach of log frequency analysis, reviewing outlier logs, and envisioning its integration into existing workflows. In addition, the intention was to get feedback to refine the approach and foster collaboration between automated techniques and human expertise for effective anomaly detection and manual root cause analysis in cloud environments.

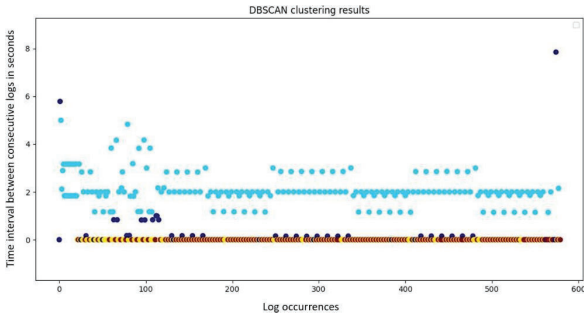
IV. RESULTS AND FINDINGS

DBSCAN clustering algorithm was implemented on the log files to identify the frequency patterns in the form of clusters. To verify that the clusters formed were accurate enough to depict the frequency patterns, hyperparameter tuning was employed to maximize the accuracy and efficiency of the algorithm and make the algorithm suitable for all log files at once. After iterative tuning of the hyperparameters, the final hyperparameters were:

- Epsilon value of 0.5
- Min_samples value of 20% of the total number of logs in each log file

Fig. 2. Logs with definite temporal patterns

- Custom distance metric: abs(x - y) / mean([x, y]), where x and y are two consecutive points in the log dataset.



A. Outcome of the timestamp-based log analysis:

After implementing the clustering algorithm, it was observed that while most of the log types displayed temporal patterns, a few log types had logs that did not adhere to a time-frequency pattern. Visual representation of some clusters is shown in Figures 2, 3, 4 where each dot on the graph represents one log in the log type. The variation in the colors of the points represents different clusters, where each cluster has logs that adhere to a specific temporal pattern. Figure 2 represents a log file which contains logs that are related to egress in a particular cloud cluster, i.e., the process where containers reach and communicate with external resources to get a particular task done in Ericsson's CNF Cloud environment. According to the clusters, the most common time intervals between the logs were recorded to be between 1-5 seconds, with a domination of 2-second time intervals. The graphical representation also shows that many logs had a time interval of 0 seconds, i.e., many logs occurred in a burst with no time difference among each other. According to domain experts, these logs are supposed to occur at regular time intervals, as the process depicted by these logs is a timed process. Thus, all the logs that fell in the red cluster (0 seconds time interval) were deduced as erroneous. There are some outlier points (depicted in navy blue in the figure) that do not follow any of the identified temporal patterns. They might be alarming, especially if it is important for the logs to occur in a timely manner. Upon being sent for further analysis, the logs associated with the outliers were found to be erroneous. This analysis was carried out as a part of the interview process. For example, the log represented by the navy blue dot on the top of the graph, which has an eight-second time interval with its preceding log, was found to be problematic after further analysis.

Figure 3 denotes a clustering result of a specific log file containing logs that do not follow any temporal pattern. The logs occur with varying time intervals ranging from 0

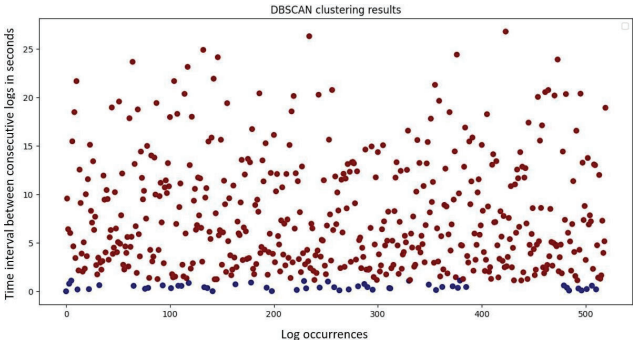


Fig. 3. Logs with no definite temporal pattern

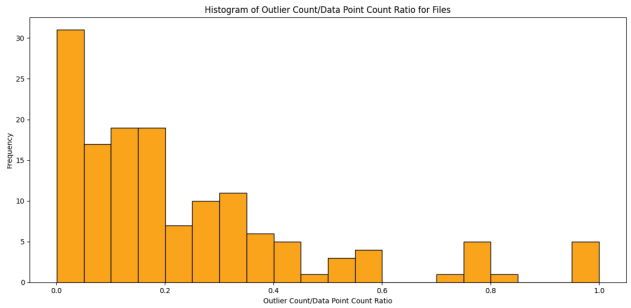


Fig. 5. Histogram representing outlier/datapoint count ratio for different log files

to 30 seconds, with most logs occurring below 15 seconds after the preceding log. However, a proper cluster depicting regular time intervals could not be formed by the clustering algorithm, as there weren't enough log occurrences that followed a regular pattern. The dots in blue represent the logs that occur immediately after their preceding log. In contrast, the dots in red represent the logs that occur after a certain time interval after their preceding log. The logs depicted in the above graph are related to a manually triggered query event. Thus, it is natural that the logs which represent this process will not occur in a timely manner.

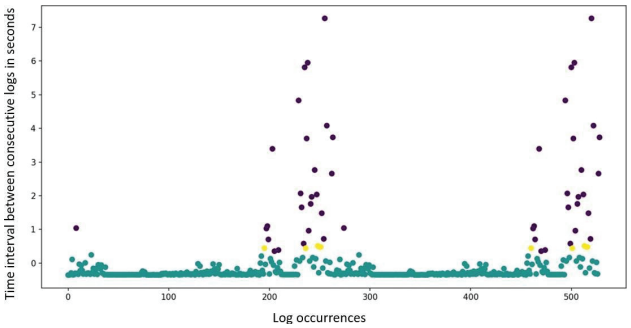


Fig. 4. Logs with temporal patterns and outliers

Figure 4 denotes a clustering result of logs that have both - temporal patterns and scattered logs. On analyzing the messages displayed by these logs, it was found that these logs belong to a process related to load balancing of a containerized application. The message displayed by the log did not show any error, but the logs were occurring continuously without any gap (represented by the sea green colored points). This immediate occurrence of a burst of logs is alarming with respect to the domain of these logs, and thus, it might indicate an error. On the contrary, the logs shown by the purple points have a consistent pattern with varying time intervals, which shows that the load balancing in the container is running well. It is noticeable from the graph that the pattern represented by the purple points repeats after 200 log occurrences. It is not necessarily a pattern with regular log occurrences, as the time differences between consecutive logs ranged from 1 second to 7 seconds in this pattern. Still, it was interesting to send the logs of this log type for further manual analysis to identify the domain of the logs and see why this pattern has occurred. In a similar manner, many other log files were analyzed using DBSCAN clustering. Here, the logs of each log type were fed to the clustering algorithm, and time patterns in each log file were identified. Most of the identified patterns helped in analyzing the logs based on their timestamps and enabled the detection of hidden errors and abnormal behavior.

Our algorithm showed promising clustering results, as it could identify temporal patterns in the logs of different log types. The best way to validate the clustering results and the overall idea of log frequency analysis was to involve human evaluation in the clustering algorithm results. As it was not possible to send all the files for manual analysis due to the limited availability of system experts and a huge number of log files, certain log files had to be selected that showed promising clusters and an optimal number of outliers. To decide which files are most suitable for further manual analysis, a few evaluation metrics, such as *outlier/datapoint count ratio* and *silhouette score* were used.

The selection criteria employed by the evaluation metrics to select the log files for outlier analysis and manual log analysis are:

- Low Mean/Std value
- Outlier/datapoint count ratio lower than 10%
- Silhouette score close to 1.0 (for files with multiple clusters)

Figures 5 and 6 provide an overview of the evaluation metrics, which helped in evaluating the clustering efficiency and choosing the files for further manual analysis. These histograms were generated to analyze the distribution of log files that fall under different values of the evaluation metrics, i.e., *outlier by data point count ratio* and *silhouette score*. The processed data set consisted of more than 200 log files, and sending all these files for manual analysis was impractical. Evaluation metrics were therefore used to select the files that can be sent for manual analysis. Having a visual representation of these metrics further facilitated a calculated selection of log files.

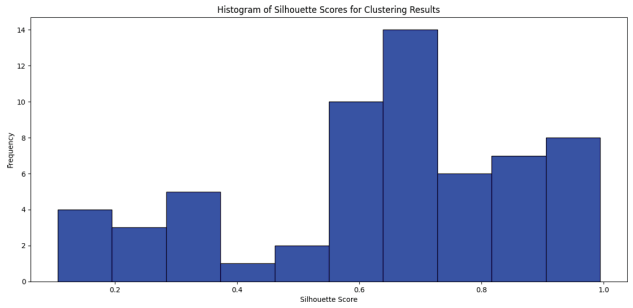


Fig. 6. Histogram representing silhouette scores of the log clusters

Figure 5 shows the distribution of the ratio of the number of outliers to the total number of logs in each file. The X-axis represents the outlier count/Data point count ratio. Here, the ratio of 1.0 indicates that all data points were outliers, and 0 indicates the absence of outliers. Keeping the log files with no outliers aside, the majority of the log files had a ratio of less than 20% (0.2 on the x-axis). Meaning that 20% of the logs in a log file were outliers. Such logs would be more appropriate to consider for anomaly detection using log frequency analysis. Therefore, we considered focusing on logs with less than 20% outliers for manual analysis.

Figure 6 shows the distribution of the silhouette score in log files containing more than one cluster. The histogram shows that the majority of the log files had a silhouette score of 0.7. However, a silhouette score close to 1 indicates distinct clusters. Hence, a combination of files that had silhouette scores of 0.7-0.9 was considered for manual analysis.

We conducted interviews involving Ericsson practitioners having experience with domain knowledge of the CNF cloud environment and manual log analysis for error detection and root cause analysis. Before the interviews, relevant log files and identified outliers were sent to the participants. Partic- ipants were provided with background information on the log analysis tool, the research on frequency analysis, and the specific context of the study.

The interview consisted of 4 practitioners with varying experiences and responsibilities to consider different perspectives on the log frequency analysis and its impact. The information of the participants is listed in Table II.

TABLE II
INTERVIEW DEMOGRAPHICS

Questions	**Participant 1**	**Participant 2**	**Participant 3**	**Participant 4**
Experience	9 Years	4 years	5 years	2 years
Role	Lead Cloud Solutions Architect	Solutions Architect	Senior Cloud Technology Developer	Cloud Technology Developer
Tasks	Higher level supervision of log-based error detection	Manual log analysis	Development of services in cloud environment	Development of Lexicon tool

B. Feedback on the timestamp-based log analysis:

The System experts, after analyzing the outlier logs of each log type, were able to find that the outlier logs indeed indicated anomalous behavior in the associated process. They felt that instead of analyzing all the logs pertaining to the failed process, only analyzing the logs that deviated from the frequency pattern reduced the effort involved in error detection by condensing the number of logs sent for root cause analysis to a great extent. This approach can also be used for proactive system monitoring and predictive maintenance of systems.

The interview structure along with the results consisting of practitioners' perceptions of the revised log analysis is provided below:

1) **Interviewee summary:** We started the interview by asking participants about their experience, roles, and the tasks they are involved in. Table II summarizes the introduction of the interviewees.
2) **Understanding the current process of Log analysis:** Participants were asked about the typical process followed to analyze logs and the problems faced while analyzing these logs for error detection analysis before proceeding with the explanation of our approach.
   Responses from the Participants: The experts analyze log messages, such as '*debug´* and '*error´* from a set of logs associated with a particular process. However, they also note that sometimes, *"the logs might not be showing any suspicious message, which makes it difficult to comprehend where the actual error is coming from"*. A Cloud Solutions Architect added that "*Another problem that we face regularly is the huge number of logs we need to analyze to get to the root cause of the error. it gets tedious sometimes.*"
3) **Explaining the revised Frequency Analysis Approach:** Participants were given a briefing about the revised log frequency analysis approach, after which they were asked if they understood the new approach, and if they perceived any benefits or limitations regarding revised the frequency analysis.
   Responses from the participants: During the interview, the participants were informed about the frequency analysis approach and its potential integration into the current workflow of the Lexicon. The solution architects who are involved in supervision of log analysis highlighted that it is not practical to manually examine the timestamps of a large number of logs. Hence, they are rarely considered in the manual analysis. However, they added that "*not looking at timestamps of the logs might lead to neglecting some logs which might be erroneous, but do not show any error message*". Furthermore, A senior cloud technology developer mentioned that "*I feel this frequency analysis approach will be beneficial in reducing the number of logs that we need to deal with for error detection because identifying timestamp-based patterns and eliminating the logs that adhere to a pattern can enable better efficiency of log analysis.*"

4) **Review of Outlier Logs:** The log files and their outliers, which were identified as a result of the clustering process, were shared with the practitioners two weeks before the interviews for them to have a thorough understanding of the logs. During the interviews, they were asked about their understanding of the domain of these logs and their experience with detecting errors using the outlier logs alone.
   Responses of the participants: Participants analyzed the log files and their corresponding outlier logs. All interviewees mentioned that they were able to get to the root cause of the error by just analyzing the outlier logs in most cases. One interviewee further elaborated that *"Out of the 5 log files provided to me, in 4 files, I could trace the error by just analyzing the outlier logs, and analyzing the temporal logs [not outliers] did not yield in the identification of extra errors."* In some cases, they could not identify the root cause but could find the error by analyzing the logs that occur immediately before and after the outlier logs, in the given cloud system environment. X added that "*the neighboring logs showed problematic behavior in the cloud environment, which led to delay of the selected [outlier] logs. In fact, this way we could catch hold of some errors which might not be visible otherwise."*
5) **Error Detection and Correlation:** Participants were asked to discuss the extent to which the outlier logs helped in error detection.
   Responses from the participants: The participants deemed that most log files' outliers were associated with errors. However, they felt that the correlation of error detection with the outliers of frequency analysis would also depend on the domain of the process depicted by the logs. For example, it is important that logs pertaining to automated processes such as health checks follow a time pattern, but the same is not true for logs depicting manually triggered events. X further elaborated "*Out of the files sent to me, one file was related to fetching data from a database deployed in one of the containers in the CNF environment. The logs mostly followed a temporal pattern, but it was not necessarily a timed process, because it can be manually triggered too. Thus, analyzing the outliers in such cases wouldn't help much, as the logs not following a time pattern in such cases don't necessarily mean that there is an error.*"
6) **Impact on Manual Analysis:** Participants were asked whether frequency analysis could reduce the manual analysis workload and how it might fit into their existing workflows.
   Responses from the participants: The participants felt that this approach would indeed help in reducing the manual effort, as in most cases, analyzing the outliers alone was enough to detect the errors. X added "*If all the logs were to be analyzed, it was a very tedious task for us. We get to the root cause eventually, but the work involved takes a lot of time. For example, one of the files which I analysed as a part of this interview, contained close to 150 logs. whereas, the outlier logs in this file were only 9. I could detect the error just by analysing these 9 logs, and tracing their neighbour logs helped me reach the root cause of the error. I checked the remaining 150 logs which followed a temporal pattern, to see if there are any underlying errors indicated by them. I could not find anything new which was not found by analysing these 9 outlier logs. So I would say this would increase the efficiency of my daily workflow to a great extent.*"
7) **Suggestions for Improvement:** Participants were asked to offer suggestions for enhancing the accuracy of the approach and additional features.
   Responses of the participants: The participants were satisfied with the frequency analysis approach and suggested integrating it into the Lexicon pipeline. A cloud technology developer further added that "*it is better to deduce a way so that you can implement it in Lexicon, which will cater to a lot of analysts, and will improve the efficiency of Lexicon in root cause analysis of errors.*"
8) **Future Considerations:** Participants were asked to share insights on potential applications beyond error detection and challenges in implementing the approach.
   Responses from the participants: During the interview, one of the participants working as a senior cloud solution architect shared an interesting use case of frequency analysis to predict maintenance needed in a cloud envi- ronment. He added "*I think since we are going to use this approach for error detection, we can go a step forward and also use this approach for predictive maintenance of the associated cloud environments. This way, we can predict arising errors before their occurrence and take proactive measures to prevent them. This will further help us, as we will not have to deal with such a huge amount of errors if they can be prevented beforehand.*"

C. Impact on manual analysis effort:

Before conducting frequency analysis, experts would have to analyze 2252 logs for anomaly detection across 20 log files. However, after performing the log frequency analysis, the number was reduced to 167 logs. These logs were identified as outliers and were helpful in tracing errors. The remaining 2085 logs, identified as normal (non-outlier) logs, did not lead to any errors in the associated processes. We found that the manual effort involved in analyzing the logs in our sample was reduced by 92.6%. This percentage is calculated by comparing the number of logs analyzed before (2252) and after (167) the introduction of log frequency analysis.

True Positives and False Positives: According to the manual analysis by system experts, 142 out of 167 outlier logs indicated anomalies, resulting in a true positive percentage of approximately 85% and a false positive percentage of approximately 15%.

V. LIMITATIONS AND THREATS TO VALIDITY

The dataset used for log frequency analysis consisted of logs that occurred in a specific time frame, which limits the results of this analysis to the sample considered in the study.

A. External Threats

The log files used for manual analysis of logs in our study were limited, as it was not possible for system experts to analyze a large amount of logs. The number of log types identified in the initial data set was ~2000, of which some types were excluded because the number of logs that these files had was not sufficient to identify frequency patterns. Out of more than 200 log files that made it to the final step of the automated analysis, 20 files were selected that could be sent to system experts for further manual analysis based on the evaluation metrics. All the files used for the manual analysis had an outlier log ratio of 10-20%, and log files that had varying percentages of outliers were not explored for manual analysis. The extent of effort reduction in manual log analysis can change with changing log data and thus cannot be generalized. However, we expect that the logs that have 10- 20% outliers based on timestamps can have similar results in terms of reduced manual effort. However, analyzing more log files manually may lead to better insights into the effectiveness of the log frequency analysis.

B. Internal Threats

The logs used in this Thesis were limited to one cloud environment of Ericsson, which makes the results of this analysis domain specific. Further research on log frequency patterns needs to be conducted to generalize the validity of this research to other domains in the field of Computing.

VI. CONCLUSIONS AND FUTURE WORK

Our experience has shown that analyzing logs for error detection and root cause analysis can be made more efficient by using a log frequency analysis approach that combines a timestamp-based and identified-based method. Collaborat- ing with system experts through interviews and manual log analysis has been instrumental in validating the efficacy of this approach. This collaborative approach has the potential to reduce manual analysis efforts and improve anomaly detection. In the future, the research could focus on refining outlier selection criteria and incorporating advanced machine learning models to automate the process of outlier extraction using log frequency analysis. The outlier logs can be identified continuously in streaming log data for the proactive identification of anomalous behavior in the associated process. This way, this analysis can be extended to all log types without restricting it to the log files that have a certain percentage of outlier logs.

REFERENCES

[1] Xavier Baril, Oihana Coustie´, Josiane Mothe, and Olivier Teste. 2020. Application Performance Anomaly Detection with LSTM on Temporal Irregularities in Logs. In *Proceedings of the 29th ACM International Conference on Information & Knowl- edge Management* (Virtual Event, Ireland) *(CIKM '20)*. Associa- tion for Computing Machinery, New York, NY, USA, 1961–1964. https://doi.org/10.1145/3340531.3412157

[2] Zhuangbin Chen, Jinyang Liu, Wenwei Gu, Yuxin Su, and Michael R. Lyu. 2021. Experience Report: Deep Learning-based System Log Analysis for Anomaly Detection. https://arxiv.org/abs/2107.05908

[3] 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* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 1285–1298. https://doi.org/10.1145/3133956.3134015

[4] Mostafa Farshchi, Jean-Guy Schneider, Ingo Weber, and John Grundy. 2015. Experience report: Anomaly detection of cloud application operations using log and cloud metric correlation analysis. In *2015 IEEE 26th International Symposium on Software Reliability Engi- neering (ISSRE)*. 24–34. https://doi.org/10.1109/ISSRE.2015.7381796

[5] Shilin He, Pinjia He, Zhuangbin Chen, Tianyi Yang, Yuxin Su, and Michael R. Lyu. 2021a. A Survey on Automated Log Analysis for Reliability Engineering. *ACM Comput. Surv.* 54, 6, Article 130 (jul 2021), 37 pages. https://doi.org/10.1145/3460345

[6] Shilin He, Pinjia He, Zhuangbin Chen, Tianyi Yang, Yuxin Su, and Michael R. Lyu. 2021b. A Survey on Automated Log Analysis for Reliability Engineering. *ACM Comput. Surv.* 54, 6, Article 130 (jul 2021), 37 pages. https://doi.org/10.1145/3460345

[7] Shilin He, Xu Zhang, Pinjia He, Yong Xu, Liqun Li, Yu Kang, Minghua Ma, Yining Wei, Yingnong Dang, Saravanakumar Rajmohan, and Qingwei Lin. 2022. An Empirical Study of Log Analysis at Microsoft. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Singapore, Singapore) *(ESEC/FSE 2022)*. As- sociation for Computing Machinery, New York, NY, USA, 1465–1476. https://doi.org/10.1145/3540250.3558963

[8] S.E. Hove and B. Anda. 2005. Experiences from conducting semi-structured interviews in empirical software engineering research. In *11th IEEE International Software Metrics Symposium (METRICS'05)*. 10 pp.–23. https://doi.org/10.1109/METRICS.2005.24

[9] Tong Jia, Yifan Wu, Chuanjia Hou, and Ying Li. 2021. LogFlash: Real-time Streaming Anomaly Detection and Diagnosis from System Logs for Large-scale Software Systems. In *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*. 80–90. https://doi.org/10.1109/ISSRE52982.2021.00021

[10] Hari Kanagala and V.V. Krishnaiah. 2016. A comparative study of K-Means, DBSCAN and OPTICS. 1–6. https://doi.org/10.1109/ICCCI.2016.7479923

[11] Steven Locke, Heng Li, Tse-Hsun Peter Chen, Weiyi Shang, and Wei Liu. 2022. LogAssist: Assisting Log Analysis Through Log Summarization. *IEEE Transactions on Software Engineering* 48, 9 (2022), 3227–3241. https://doi.org/10.1109/TSE.2021.3083715

[12] Tarannum Shaila Zaman, Xue Han, and Tingting Yu. 2019. SCMiner: Localizing System-Level Concurrency Faults from Large System Call Traces. In *2019 34th IEEE/ACM Interna- tional Conference on Automated Software Engineering (ASE)*. 515–526. https://doi.org/10.1109/ASE.2019.00055

[13] Pengpeng Zhou, Yang Wang, Zhenyu Li, Xin Wang, Gareth Tyson, and Gaogang Xie. 2020. LogSayer: Log Pattern-driven Cloud Component Anomaly Diagnosis with Machine Learning. In *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*. 1–10. https://doi.org/10.1109/IWQoS49365.2020.9212954