FLAP: An End-to-End Event Log Analysis Platform for System Management

Tao Li
Nanjing University of Posts and Telecommunications
Florida International University
taoli@cs.fiu.edu

Yexi Jiang, Chunqiu Zeng
Computing and Information Sciences
Florida International University
Miami, USA
{yjian004,czeng001}@cs.fiu.edu

Bin Xia, Zheng Liu
Computer Science
Nanjing University of Posts and Telecommunications
Nanjing, China

Wubai Zhou, Xiaolong Zhu, Wentao Wang
Computing and Information Sciences
Florida International University
Miami, USA

Liang Zhang, Jun Wu, Li Xue, Dewei Bao
Huawei Nanjing Research and Development Center
Nanjing, China

ABSTRACT

Many systems, such as distributed operating systems, complex networks, and high throughput web-based applications, are continuously generating large volume of event logs. These logs contain useful information to help system administrators to understand the system running status and to pinpoint the system failures. Generally, due to the scale and complexity of modern systems, the generated logs are beyond the analytic power of human beings. Therefore, it is imperative to develop a comprehensive log analysis system to support effective system management. Although a number of log mining techniques have been proposed to address specific log analysis use cases, few research and industrial efforts have been paid on providing integrated systems with an end-to-end solution to facilitate the log analysis routines.

In this paper, we design and implement an integrated system, called *FIU Log Analysis Platform* (a.k.a. FLAP), that aims to facilitate the data analytics for system event logs. FLAP provides an end-to-end solution that utilizes advanced data mining techniques to assist log analysts to conveniently, timely, and accurately conduct event log knowledge discovery, system status investigation, and system failure diagnosis. Specifically, in FLAP, state-of-the-art template learning techniques are used to extract useful information from unstructured raw logs; advanced data transformation techniques are proposed and leveraged for event transformation and storage; effective event pattern mining, event summarization, event querying, and failure prediction techniques are designed and integrated for log analytics; and user-friendly interfaces are utilized to present the informative analysis results intuitively and vividly. Since 2016, FLAP has been used by Huawei Technologies Co. Ltd for internal event log analysis, and has provided effective support in its system operation and workflow optimization.

KDD '17, August 13-17, 2017, Halifax, NS, Canada

DOI: 10.1145/3097983.3098022

CCS CONCEPTS

•**Information systems** →**Data analytics;** •**Networks** →*Network monitoring;* •**Computing methodologies** →*Machine translation;*

KEYWORDS

Data Mining System, Log Processing, Event Mining, Event Summarization

1 INTRODUCTION

Modern systems and applications are continuously generating many kinds of events, from low level events such as system utilization and network traffic events, to high level events such as HTTP requests and UI click events. These events capture the systems' behaviors and contain the clues of potential operation issues. Intuitively, if the recorded events can be fully utilized, the downtime of the systems can be minimized, and the system performance and operation cost can also be optimized.

The necessity of effective event log analysis can be reflected by the real *Comcast* (i.e., the largest broadcasting and cable television company in the U.S.A) service outage event. In February 2016, users frequently encountered various types of problems when using the services of Comcast's Internet (e.g., Xfinity or Internet-based television). Although the website is integrated with a comprehensive logging system that records everything about the system status, the tech team still failed to troubleshoot the service outage in time. Due to the inability of quick problem diagnosis, the services were interrupted for hours and more than 27 million people across the country were impacted[1].

Besides *Comcast* service outage event, a number of other similar system failure cases in recent years also resulted in serious consequences[2].Therefore, people gradually pay increasing attention to advanced event log analysis techniques, hoping that the problem pinpoint efficiency can be improved and the failure can be effectively identified or avoided before it causes grave consequences.

[1]http://money.cnn.com/2016/02/16/technology/comcast-outage-over/index.html
[2]http://abcnews.go.com/Technology/facebook-social-network-suffers-outage/story?id=24808635

Recently, a lot of research efforts have already been focused on facilitating log processing [12, 13]. Many tools have been developed to address event log management for specific use cases, such as event monitoring [18], event archiving [2], event information extraction, and event querying [4, 26]. Also, a few new companies, such as *Splunk*[3], *AppFirst*[4], and *Loggly*[5], have been founded to provide convenient log management products. Traditional IT giants, such as *HP*, *IBM*, and *Amazon*, have also developed their own internal log management systems, such as *HP OpsAnalytics* [7], *IBM Tivoli* [8], and *Amazon CloudWatch* [1].

These products greatly facilitate system administrators' daily tasks, but they are still inadequate to provide a generic, comprehensive, and end-to-end solution. On one hand, the products developed by the log management companies are generic and efficient, but their analytic modules still have much room for improvement. For example, *Splunk* is able to collect the logs from the machines and is Operating System independent. It has superior log processing capabilities in terms of scalability and performance, thus enabling simultaneously log collection and indexing from a large number of machines. However, this product only provides the most fundamental analytics tools such as dashboards with basic statistics and keyword based search. These products are adequate for daily inspection but would not be helpful enough for some more complex tasks such as failure detection and diagnosis. On the other hand, the products developed by the IT services companies are highly customized and are equipped with more advanced analytics tools. However, these products are not generic as they are often adherent to their own IT services. For example, *HP OpsAnalytics*, a comprehensive product that provides advanced data analytic tools, is able to help system administrators to quickly conduct predictive analysis and failure diagnosis. Whereas, *HP OpsAnlytics* is part of the HP service ecosystem, and it is not trivial to be integrated with other non-HP IT services.

1.1 Challenges and Proposed Solutions

Considering the increasing scalability and complexity of event log data, designing and developing a generic event analysis system is not a trivial task. Based on the observations we have made during our preliminary study, we have identified several key challenges and proposed an end-to-end solution called *FIU Log Analysis Platform* (FLAP) to address the challenges.

Challenge 1. *Given various types of event logs, how to support event analysis in a generic and comprehensive manner?*

Existing log management products, like the ones we mentioned above, do not provide a generic and comprehensive solution. The products developed by log management companies are able to handle various types of event logs, but they are less comprehensive due to the lack of advanced data mining based solutions. The products developed by IT services companies provide complete functions from log collection to log analysis. However, such products are tailored to specific ecosystems that cannot be easily adapted.

FLAP is designed to be both generic and comprehensive. In terms of generality, the data processing and event mining libraries in FLAP are designed to be domain independent. Such a design makes FLAP to be flexible enough to handle various types of logs and to conduct various types of event analysis tasks. Although, FLAP has already included a number of data pre-processing and event mining algorithms, users can still import new algorithms into the system to enrich the power of the system. In terms of comprehensiveness, FLAP supports the entire workflow of event analysis, including log pre-processing, event extraction, event querying, event mining, and event visualization. There is no need to use a third-party tool during the whole event analysis procedure.

Challenge 2. *Facing various analysis requirements with different objectives, how to effectively adapt existing analysis methods for highly customized analysis tasks?*

In FLAP, Challenge 2 is effectively addressed by developing appropriate data mining algorithms for log analysis. Specifically, three problems are considered to be important and critical to the system administrators: 1) Discovering the correlations and casuality among different event types; 2) Summarizing and demonstrating the running status of the target systems; 3) Forecasting the potential problems and troubleshooting the failures of the systems.

To solve the aforementioned problems, we design a number of advanced pattern mining algorithms and integrate them into FLAP, including event temporal dependency mining, event temporal lag mining, and failure prediction. Leveraging these algorithms, system administrators can effectively discover a various types of correlations among the events and obtain insightful information about the systems.

Challenge 3. *Given various types of analysis results, how to effectively present them to users?*

Presenting the data as well as the analysis results in a reasonable and intuitive way is not a trivial task. Typically, the raw data and results can be easily understood by data mining experts, but they are less informative to the people without technical background. To address this challenge, FLAP provides a user-friendly dashboard with intuitive charts and diagrams. Moreover, to facilitate data exploration, FLAP supports *Dynamic Query Form*, an intelligent mechanism for users to perform interactive data exploration.

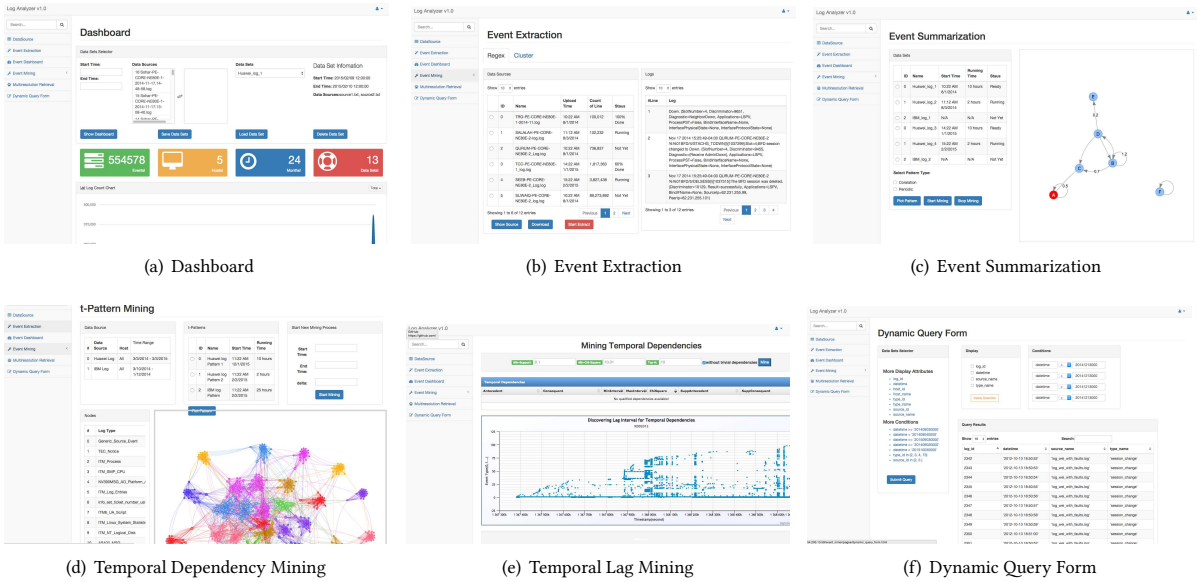
1.2 Roadmap

The rest of this paper is organized as follows. Section 2 presents an overview of FLAP. In Section 3, we discuss how the logs are preprocessed and introduce the log organization strategies supported in FLAP. In Section 4, we describe how FLAP conducts event analysis via multi-resolution analysis, event summarization, event pattern mining, and system failure prediction. In Section 5, we introduce the visual component of FLAP and demonstrate its information visualization capability. Section 6 presents the system evaluation as well as the case study. In Section 7, we introduce the related work. Finally, Section 8 concludes the paper.

2 FLAP OVERVIEW

FLAP is a web-based integrated system for log analysis. The screenshots of FLAP are shown in Figure 1 and the architecture overview is displayed in Figure 2. Using FLAP, users can conduct different stages of event processing/analysis seamlessly and conveniently.

[3]http://www.splunk.com/
[4]http://www.appfirst.com/
[5]https://www.loggly.com/



(a) Dashboard (b) Event Extraction (c) Event Summarization

(d) Temporal Dependency Mining (e) Temporal Lag Mining (f) Dynamic Query Form

Figure 1: The screenshots of FLAP.

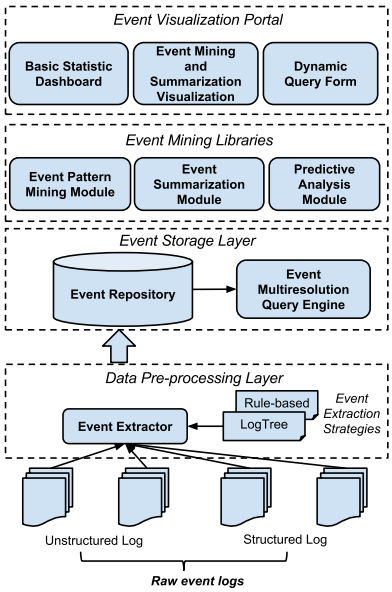


Figure 2: System architecture.

In general, FLAP consists of four layers of components, including *Data Pre-processing Layer*, *Event Storage Layer*, *Event Mining Libraries*, and *Event Visualization Portal*.

Event Pre-processing Layer provides a flexible way to conduct all pre-processing of the event logs. Due to the diversity of event log formats and event data sources, it is not likely to design and implement a one-fit-all pre-processing module to effectively handle all types of event logs. To be flexible, this module is designed to be extensible and currently includes two log pre-processing approaches: *rule based event extraction* and *template learning*. The first approach is the traditional way for domain experts to define the ad-hoc rules to extract the events from unstructured or semi-structured logs. The second approach is a more advanced approach for event extraction. It leverages unsupervised learning techniques to automatically learn the event templates and then extract the events accordingly. This approach can be used when the number of latent event types is too large to be extracted via manually set rules.

Event Storage Layer handles the storage and data retrieval tasks. Once the logs are pre-processed, the extracted events will be stored in a specified repository. To balance both space cost and retrieval efficiency, we carefully designed the storage format of the events.

Event Mining Libraries is the core part of the system. It integrates a number of useful event processing and mining algorithms for different use cases, including event multi-resolution analysis, event summarization, event pattern mining, and system failure prediction. These algorithms can be used to help system administrators for system diagnosis.

Event Visualization Portal is the interface between the system and the users. To intuitively present the analysis results, this layer is designed to be user-friendly and easy to operate. Specifically, a dashboard is provided to illustrate the basic statistics; Dynamic Query Form is presented to enable the interactive and intelligent event data exploration; Summarization and Visualization module is presented to display the mining results.

3 LOG ORGANIZATION

Usually, the raw log messages only contain the unstructured or semi-structured text instead of the well-defined events. Figure 3 (a) shows an example of raw log message obtained from FLAP. Discrete or structured events are much easier to be visualized and explored by human experts than raw log messages. Therefore, there is a need to convert the raw logs into discrete or structured events. The *Event Pre-processing Layer* of FLAP provides two solutions to event extraction: *Rule Based Event Extraction* and *Unsupervised Template Learning*.

3.1 Rule Based Event Extraction

Rule based event extraction is a traditional way that allows people to manually define the regular expression rules to extract the events. In FLAP, system administrators can create extraction configuration files, and each file would contain multiple extraction rules that can be applied to extract the events from raw log messages. Concretely, there are four components for each rule: *event name*, *type pattern*, *extraction pattern*, and *parameters*. In particular, *event name* defines the name of the extracted event; *type pattern* defines the patterns to match the messages; and *extraction pattern* and *parameters* define the parameters that need to be extracted. An example rule can be seen in Figure 3 (b).

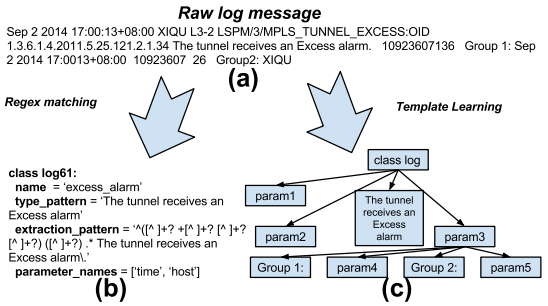


Figure 3: Regex matching and template learning.

For each raw log message, the first matched rule will be used and the corresponding parameters of the events will be extracted. The extracted events are then would be stored and used by downstream analysis tools.

Rule based event extraction is an intuitive method to extract events from raw logs. This method is useful if the number of latent types is small and the types can be easily identified. But, it is common that logs can have a large number of event types in practice. For example, *Windows Event Manager* typically records events with thousands of types [10]. Many Linux services, such as *openssh*, *Kerboros 5*, and *Samba 3* [27], also have similar number of event types. Therefore, to handle complex logs, machine learning and data mining based methods are needed.

3.2 Learning Based Event Extraction

When the number of latent event types in the raw logs is large, the time cost for rule based event extraction becomes prohibitive. According to our experiences, a moderately complex log would require 2 days for a domain expert to define the rules.

To deal with complex raw textual logs, we apply template learning for automatic event extraction [21]. In general, template learning utilizes the format and structural information of the raw logs in the clustering process to generate events. An example of the extracted template, is shown in Figure 3 (c). In template learning, the problem of discovering the event types is formulated as follows: Given a set of messages $S = \{s_1, s_2, \cdots, s_n\}$, the objective of the template learning is to find a representative set of $k \leq n$ log messages, named S^*, to express the information of S as much as possible, where each element in S^* represents one type of event, and k is a user-defined parameter. To quantify the goodness of S^*, the *event coverage* ($J_C(S^*, S)$) of S^* is used as the objective function and its form is defined as follows:

$$\max J_C(S^*, S), \text{ subject to } |S^*| = k,$$
$$\text{where } J_C(S^*, S) = \sum_{x \in S} \max_{x^* \in S^*} F_C(x^*, x). \tag{1}$$

In Equation (1), $F_C(x^*, x)$ is the similarity function of the log message x^* and the log message x.

To measure the similarity between log messages, a message is transformed into a tree $T = \{V, E, L, v_{root}, P\}$, where V is the set of nodes, E is the set of edges, P is the set of log message segments, L is the mapping function between the set of nodes and the set of log messages segments, i.e. $L : V \rightarrow P$, and v_{root} is the root node. Given two messages s_1 and s_2 and their corresponding trees $T_1 = \{V_1, E_1, L, r_1, P\}$ and $T_2 = \{V_2, E_2, L, r_2, P\}$, the similarity is quantified as follows:

$$F_C(s_1, s_2) = \frac{F'_C(r_1, r_2, \lambda) + F'_C(r_2, r_1, \lambda)}{2}, \text{ where}$$

$$F'_C(v_1, v_2, w) = w \cdot d(L(v_1), L(v_2)) + \sum_{(v,u) \in M^*_C(v_1, v_2)} F'_C(v, u, w \cdot \lambda).$$

In the above equation, λ is a user-defined parameter between 0 and 1. $M^*_C(v_1, v_2)$ denotes the best matching between the children of v_1 and v_2. The function $d(\cdot, \cdot)$ measures the similarity between two log message segments. Given two message segments: $m_1 = p_1, \cdots, p_{n_1}$ and $m_2 = q_1, \cdots, q_{n_2}$, their *similarity* is computed in Equation (2).

$$d(m_1, m_2) = \frac{1}{\sqrt{n_1 \cdot n_2}} \sum_{i=1}^{\min(n_1, n_2)} x_i \tag{2}$$

By leveraging *message segment table* [21], the event template can be learned using clustering and the whole procedure can be conducted in $O(n^2)$ time in the worst case, where n is the number of log messages.

4 EVENT ANALYSIS

The main part of FLAP, *Event Mining Libraries*, provides a large number of efficient and effective event processing and mining algorithms and greatly facilitates administrators managing and maintaining the modern systems.

4.1 Multi-resolution Event Exploration

Exploring useful knowledge from the logs depends on the choice of data granularity, time range, and the event type set. For different people with different purposes, there are various ways of conducting event analysis. Before finding insightful information, system administrators would conduct multiple rounds of analysis using different analysis algorithms/parameters on different data inputs.

To facilitate the repetitive event analysis, FLAP develops a multi-resolution data exploration module that decouples the procedures of data preparation and data analysis [9]. Using this module, users can easily compose various event analysis tasks and then efficiently execute them without redundant data preparation. To integrate this module into FLAP, we modified the implementation and made it adaptable to all downstream event analysis algorithms.

With the support of multi-resolution exploration, FLAP is quite flexible where many real-life scenarios can be adequately and efficiently handled and supported. Three typical use cases of FLAP are described and discussed as follows.

Scenario 1. *A whole year log of the target system is available in* FLAP*, but the system administrator only wants to analyze the events that occurred during the latest 1 month. Moreover, she/he is not interested in the events that are related to "firewall scan".*

Scenario 2. *Specific details contribute little to the investigation of long term system behaviors. Moreover, analyzing logs with large numbers of records is time-consuming for computational intensive data mining algorithms. Therefore, it is preferable to perform analysis with the needed details at the appropriate level of granularity.*

Scenario 3. *After conducting event summarization, the system administrator found that one particular time period of events is suspicious. Therefore, she/he intends to use other event mining techniques to find out more details.*

Typically, performing each of the task in the aforementioned scenarios requires non-trivial efforts from the system administrators. This is because they need extra efforts to pre-process the data before each trial, and different tasks require different kinds of pre-processing. The multi-resolution data exploration module in FLAP provides carefully designed event operators to handle the above tasks. Generally, there are two groups of operators for event manipulation: the *data transformation operators* and the *data query operators*. The first group of operators contains six operators, including *Vectorize/Unvectorize*, *Encode/Decode*, *Prune*, and *Concatenate*. They are used to transform data from one type to another. The second group of operators contains four operators, including *Project*, *Select*, *Zoom*, and *Describe* (indicating the concrete analysis algorithm). They are used to retrieve event data with different query conditions.

Based on the definition of these proposed operators, multi-resolution event manipulation can be described as the following expression:

$$\Upsilon_{name}(\sigma^*_{[t_{start},t_{end}]}\tau^*_u\Pi^*_{E\in\mathcal{P}(\mathcal{E})}(\mathcal{F})),$$

where Υ_{name}, i.e. *Describe* operation, indicates the name of the concrete analysis algorithm, $\sigma_{[t_{start},t_{end}]}$, τ_u, and $\Pi_{E\in\mathcal{P}(\mathcal{E})}$ represent *Select*, *Zoom*, and *Select* operation respectively, and symbol $*$ denotes conducting the operation 0+ times. This expression shows that the events can be extracted with any event type subset in any resolution during any time range with any analysis algorithm.

4.2 Temporal Dependency Mining

Discovering interesting patterns from sequential data types can provide great insights about the system behaviors. For example, a computer system problem may trigger a series of symptom events, indicating a natural signature for identifying the root cause of system problems. Traditionally, mining frequent episodes from an event sequence can be attained by predefining a fixed time window size. With the help of window size, items in the same sliding window are viewed as an itemset in a single transaction. Then the idea of mining frequent itemsets from transaction data is applied for discovering frequent episodes. However, this method causes two issues which required to be solved in real applications.

Issue 1. *The fixed time window scheme can not explore temporal information within a window precisely, and misses the opportunity to discover the temporal relationship with time distances larger than the predefined time window size.*

Issue 2. *The frequent pattern mining framework incurs a well-known issue that it fails to discover the infrequent but significant patterns which often acquire more concern in some scenarios such as system management. For example, in a well-managed system, frequent patterns are normal operations and service disruptions are usually infrequent but significant patterns.*

The above issues can be addressed by temporal dependency pattern mining. The problem setting of temporal dependency pattern mining can be briefly described as follows: Given the occurrences sequence of two event types $S_A =< a_1, a_2, ..., a_j, ..., a_m >$ and $S_B =< b_1, b_2, ..., b_i, ..., b_n >$, a temporal (dependency) pattern defined over S_A and S_B is denoted as $A \rightarrow_{[\tau-\delta,\tau+\delta]} B$, which indicates B statistically depends on with time period τ and time variance δ. To discover the temporal patterns in event sequences, FLAP integrates the t-pattern method [14], which is able to effectively identify the dependencies between two types of events without introducing time windows.

To discover the dependency, two types of distributions are evaluated and they are defined as follows. ***Unconditional Distribution***. The unconditional distribution of the waiting time for event B is defined as

$$F_B(r) = P(d(x, S_B) \leq r),$$

where $x \in R, r \in R$ and $d(x, S_B)$ is the distance between a timestamp x and a sequence of time stamps S_B, defined as

$$d(x, S_B) = \inf_{z\in S_B \wedge z\geq x} ||z - x||_{\ell 1}.$$

$F_B(r)$ describes the probability of having event B occurring within an time interval r after any timestamp x.

The unconditional distribution of the waiting time for event B can be estimated from the observed event sequence $S_B =< b_1, b_2, ..., b_i, ..., b_n >$, where $1 \leq i \leq n$. For example in Figure 4 (a), the unconditional distribution $F_B(r)$ of the waiting time for event B are estimated as $F_B(r) = P(t_B - t_x < r) = \frac{len(\{x\in[0,T]:d(x,S_B)\leq r\})}{len([0,T])}$. Thus, $F_B(10) = \frac{5+10+10+10+10}{90} = 50\%$.

Conditional Distribution. The conditional distribution of the waiting time for event B with respect to event A is defined as

$$F_{B|A}(r) = P(d(x, S_B) \leq r : x \in S_A),$$

where $F_{B|A}(r)$ describes the conditional probability distribution given an event A happening at timestamp x and it is estimated as: $F_{B|A}(r) = \frac{1}{m}|\{a_j : d(a_j, S_B) \leq r \wedge a_j \in S_A\}|$. Thus, in Figure 4 (b), $F_{B|A}(10) = \frac{3}{4} = 75\%$.

Therefore, given two event sequences S_A and S_B, $A \rightarrow B$ iff $F_B(r)$ is significantly different from $F_{B|A}(r)$. As shown in Figure 5, significant difference between unconditional and conditional distribution is illustrated when two events are dependent, while no obvious difference is presented given two independent events.

The dependency test between A and B is based on the comparison between the first moments of $F_B(r)$ and $F_{B|A}(r)$ respectively. M_B and $M_{B|A}$ can be computed as $M_B = \frac{1}{2T}\sum_{i=1}^{n} t_i^2$ and $M_{B|A} = \frac{1}{m}\sum_{j=1}^{m} d_j$. Under the null hypothesis that A and B are independent, the strength of their dependency can be evaluated using statistical testing [14].

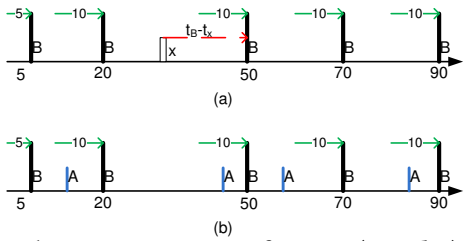


Figure 4: There are two types of events (A and B) happening during the time interval $[0, 90]$. Time lag $r = 10$ is given. (a) shows $F_B(r)$ describing the probability that any random point x followed by an event B within time lag r. (b) shows $F_{B|A}(r)$ denoting the possibility that A is followed by B within time lag r.

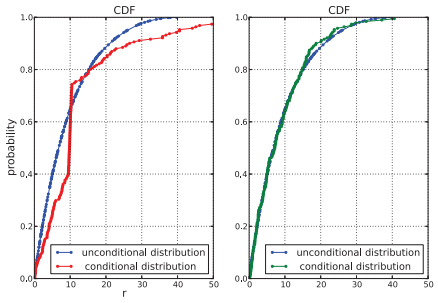


Figure 5: Cumulative probability distribution function. Unconditional and conditional distributions for two dependent events are shown in the left subfigure, while the right subfigure displays the unconditional and conditional distribution for two independent events.

4.3 Event Temporal Lag Mining

Time lag is one of the key features to depict the temporal dependencies of the events. It is informative to help discover the evolving trends of the coming events and predict the future behavior of the systems. In FLAP, an effective event lag mining algorithm is developed to help system analysts to quickly identify the latent temporal correlations between event pairs [28].

Suppose $S_A =< a_1, \cdots, a_m >$ and $S_B =< b_a, \cdots, b_n >$ are two sequences of events with types A and B, the time lag between event A and B ($A \rightarrow_L B$) is modeled as the time interval μ between the most appropriate event pairs a_i and b_j with noise ϵ, i.e. $b_j = a_i + \mu + \epsilon$.

In order to discover the temporal dependency rule $A \rightarrow_L B$, we need to learn the distribution of the lag L. The problem of identifying the most likely time lag between event pairs is equivalent to learn the parameter Θ that determines the distribution of the time lag L. The intuitive idea of solving this problem is to find the most likely parameter Θ given both sequences S_A and S_B, as quantified by Equation (3).

$$\hat{\Theta} = \arg\max_{\Theta} P(\Theta|S_A, S_B), \tag{3}$$

which can be further reduced to

$$\hat{\Theta} = \arg\max_{\Theta} \ln P(S_B|S_A, \Theta), \tag{4}$$

by applying the Bayes Rule, taking the logarithm on both sides, and eliminating the irrelevant terms.

By assuming the event instances in S_B are mutually independent given the sequence S_A and value of Θ if B is caused by A, we can get

$$P(S_B|S_A, \Theta) = \prod_{j=1}^{n} P(b_j|S_A, \Theta). \tag{5}$$

As L is a latent constant and ϵ is a random variable, the distribution of L can be expressed as a normal distribution with mean μ and variance σ, i.e. $L \sim \mathcal{N}(\mu, \sigma^2)$.

To estimate Θ, the EM algorithm is applied with *Expectation* quantified as Equation (6) and *Maximization* quantified as Equation(7).

$$\text{Expectation: } r_{ij} = \frac{r'_{ij} \times \mathcal{N}(b_j - a_i|\mu', \sigma'^2)}{\sum_i^m r'_{ij} \times \mathcal{N}(b_j - a_i|\mu', \sigma'^2)} \tag{6}$$

$$\text{Maximization: } \begin{cases} \mu = \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{m} r_{ij}(b_j - a_i) \\ \sigma^2 = \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{m} r_{ij}(b_j - a_i - \mu)^2 \end{cases} \tag{7}$$

The optimization can be easily resolved using standard EM optimization method [28].

4.4 Event Summarization

Section 4.2 and Section 4.3 introduced two concrete mining techniques in *Event Mining Libraries* of FLAP: temporal dependency mining and event temporal lag mining. Both techniques are used for efficiently uncovering the hidden information from dynamic systems. However, as the size of event logs grows dramatically, most pattern mining techniques would typically output a large number of patterns that would overwhelm system administrators. Hence, it is often preferable to have a global overview of the observed system before conducting the detailed system analysis [12, 25].

In FLAP, we developed *Natural Event Summarization* to summarize the events from the temporal perspective. *Natural Event Summarization* is able to summarize the given events using inter-arrival histograms in order to capture the temporal relationships among events [10]. It leverages a two-stage optimization guided by the *Minimum Description Length Principle* [6] that compromises the summary results between accuracy and brevity. A high level overview of the procedure of natural event summarization is illustrated in Figure 6.

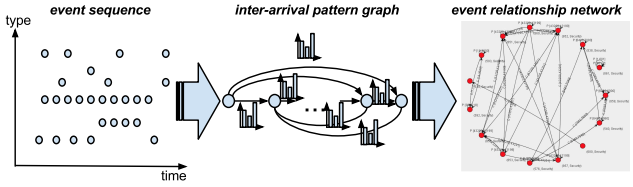


Figure 6: Event summarization.

Suppose an event sequence D is the event sequence with n event instances in form of event-timestamp pairs, i.e. (t, e) and $D =$ ($< t_1, e_1 >, \cdots, < t_n, e_n >$), the problem of *Natural Event Summarization* can be formulated as follows: Given an event sequence D, for each subsequence S containing event types x and y,

the minimum description length $L(S)$ for S is defined as

$$L(S) = \arg\min_{\{S_1, S_2, \cdots, S_k\}} \sum_i L(h(S_i)). \quad (8)$$

In Equation (8), S denotes the subsequence of D that only contains events of types x and y, S_i denotes the disjoint segment of S, i.e. $S = (S_1, S_2, \cdots, S_k)$, and $L(h(S_i))$ denotes the minimum description length of S_i.

The minimum description length of $L(h(S_i))$ is quantified according to Equation (9), which is quantified by the number of bits to describe the corresponding histogram of S_i.

$$L(h(S_i)) = \arg\min_{\bar{h}(S_i) \in \bar{H}(S_i)} L(\bar{h}(S_i)) + L(h(S_i)|\bar{h}(S_i)). \quad (9)$$

The description length $L(h(S_i))$ consists of two parts: The description length of the closest standard histogram $L(\hat{h}(S_i))$ to the histogram of S_i and the description length depicting the differences between the histogram of S_i and the corresponding standard histogram. The first part is quantified by considering multiple factors of the standard histogram, such as the number of event types in D, boundary locations of S_i, number of non-empty bins in the histogram, etc. The second part is quantified from the perspective of histogram transformation in terms of bin movements [3].

4.5 Failure Prediction

Successful prediction of failures offers the promise of enabling system self-configuration and self-management.Such predictions can help reduce system maintenance cost to avoid unplanned outages. Even if the prediction of an error happens too late to allow proactive action, it can prevent the spread of the error to the entire network and can also be used as a foundation for efficiently identifying root causes.

FLAP supports failure prediction for log based problem diagnosis and determination. Specifically, we incorporate a simple yet effective failure prediction method that is able to identify the potential failures using supervised learning method [15]. In FLAP,

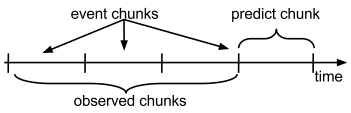


Figure 7: Illustration of failure prediction.

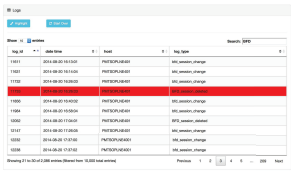
failure predication is conducted as a four-stage approach. First, the events in logs are chunked using sliding windows. For each sliding window, the events are further divided into several event chunks, each of which contains the events occurred within a specified time range (e.g. 1 minute). Suppose there are n chunks, the first $n-1$ chunks are considered as the observed chunks and the last chunk is treated as the predict chunk (See Figure 7). Then, various groups of features are extracted from each chunk of a given sliding window. For example, the number of events of each severity level occurs in each chunk, such as *INFO*, *WARNING*, *ERROR*, *FATAL*; the number of events occurred in each chunk for each event type; the accumulated number of events occurred for each event type; whether there exists *FATAL* events in observed chunks. Once the features are extracted, data normalization is then conducted to eliminate the scale inconsistency of the features. Having prepared the data, state-of-the-art classification models such as *SVM*, *Ridge Logistic Regression*, and *Lasso* are used to conduct the predictive analysis. Finally, the majority vote principle is leveraged to aggregate the individual models to generate the final prediction results.

As failure prediction is naturally an imbalanced two-class classification problem, the positive (failure) samples are much less than the negative samples. To address this problem, we leveraged cost-sensitive classification techniques to balance the importance of positive and negative samples.

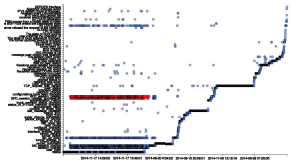
5 EVENT VISUALIZATION

In addition to offer conventional visualization tools, such as event bar chart, event cube, and 2D/3D event graph, the *Event Visualization Portal* of FLAP provides two useful ad-hoc analysis tools: dashboard and dynamic query form.

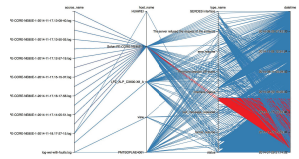
5.1 Dashboard



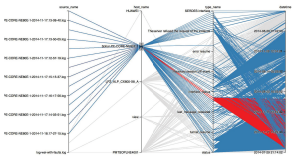
(a) Log List



(b) Scatter Plot for Different Types of Event



(c) Parallel Coordinate for the Selected Dataset



(d) Parallel Coordinate for a Specific Host (Sohar-PE-CORE-NE80E-1)

Figure 8: Event Data Visualization.

Dashboard is the most intuitive and fundamental way to provide an overview of the managed logs. In FLAP, the integrated dashboard (See Figure 1(a)) illustrates several basic statistics about the selected event log, including the number event instances, the number of hosts that generate the events, time range of the events, etc. Additionally, various types of charts (See Figure 8) are utilized to provide a general overview of the events, such as the time series reflecting the volume of each time slot and the pie chart reflecting the constituent of the event types.

All the above information is integrated into a concise report that gives the system administrators a quick summary of the selected event log. The knowledge in the report can be understood in a few minutes by the administrators. It is sufficient for daily inspection if the recorded system is in normal condition.

5.2 Dynamic Query Form

Besides the dashboard, a more advanced data exploration module called *Dynamic Query Form (DQF)* is also provided in FLAP to enable system administrators actively retrieve the results they need [22].

The essence of *DQF* is that *DQF* can capture users' interests during the interactions and actively recommend the most relevant query component to the users. As event analysis is an iterative procedure, *DQF* is a suitable tool to ease the analysis tasks for the administrators.

Building on top of the multi-resolution event exploration module, *DQF* is able to generate the query form that satisfies the following template:

$$F = \text{SELECT } A_1, A_2, \cdots, A_k \\ \text{FROM } \mathcal{R}_F \text{ WHERE } \sigma_F \\ \text{WITH RESOLUTION } r, \tag{10}$$

where $\mathcal{A}_F = \{A_1, A_2, \cdots, A_k\}$ are k attributes for projection; R_F denotes the data source of the events; σ_F denotes the conjunction of expressions for selection like the event types and the time range; and r denotes the time resolution of the events.

6 SYSTEM EVALUATION AND CASE STUDY

The system evaluation is conducted on two levels: the module evaluation and the overall system evaluation. The module evaluation is based on standard performance metrics used in the corresponding research literature. The evaluation results demonstrate the efficacy and effectiveness of the proposed system in the aspects of event analysis, pattern discovery, and problem diagnosis.

6.1 Case Study

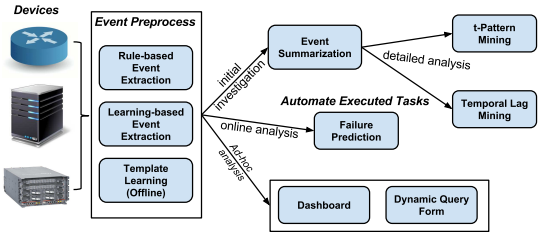


Figure 9: Log event analysis workflow using FLAP.

Figure 9 illustrates the typical workflow of log event analysis using FLAP. The system is able to obtain the event logs in both active and passive manners. The *active* manner is mainly for offline analysis. It requires the system administrators to specify the location of the logs, typically stored in an accessible file system. The system will fetch the dataset and conduct the event preprocess and analysis. For the *passive* manner, the system administrators need to set up the message passing service to periodically send log messages to the specific port of FLAP. In this case, FLAP is able to receive and conduct continuous analysis.

In the case study, we use three concrete tasks to demonstrate how system administrators can leverage FLAP to inspect an internal network of Huawei Technologies Co. Ltd. with event logs. For the confidential issue, we named the network as *X*. The experimental environment is shown in Table 1 and the basic statistics of the event log is available in Table 2. Due to the scalability of the network, only the significant events such as important operations, warnings/errors, and periodical background tasks are recorded.

Table 1: Experimental Environment

OS	bits	Memory	CPU
Linux 2.6.32	64	32G	Intel(R) Xeon(R) CPU E5-2430L v2 @ 2.40GHz, 6 core

Table 2: Event log dataset.

Data Source	Network X
# Event Types	90
# Daily Significant Events	299,815

Task 1: Event Summarization. It is important for system administrator to have a global overview of the observed system before performing the detailed system log analysis. The event summarization is conducted on the dataset listed in Table 2. Figure 10 shows the ERN(Event Relationship Network) generated by the event summarization module. ERN shown in the figure are immediately useful for managing computer systems by system administrator. As Figure 10 shows, the events like login, logout, BFD_session_change, and bfd_session_created_successfully are triggered at higher frequency than other events.

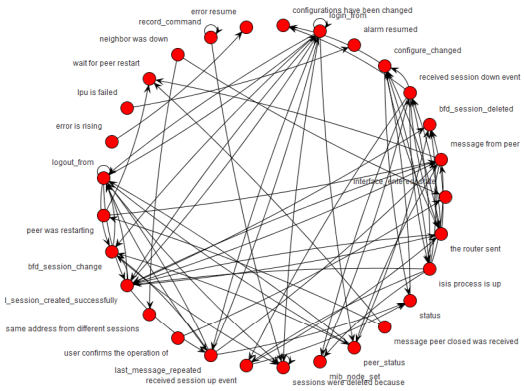


Figure 10: Event Relationship Network Generated by Event Summarization

Task 2: Root cause mining. The domain experts already have the knowledge about which events are critical and can directly result in failure (Some example of raw messages are listed in Table 3). However, it is not obvious to identify the causes that would trigger the occurrences of these critical events.

Table 3: Examples of critical log messages for *Network X*.

Type No.	Example Message
16	`The line protocol None on the interface Tunnel0/0/11 has entered the DOWN state.`
132	`Sessions were deleted because the session hold timer expired and the notification of the expiry was sent to the peer.`

To find out the possible causes, the temporal dependency pattern mining module (See Figure 1(e)) is used to find out all the correlated event pairs $A \rightarrow B$, where B is considered as the candidate critical events. For the parameter settings, the time tolerance δ is set as 3

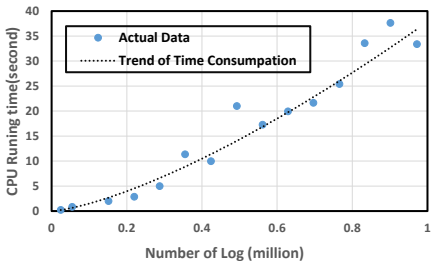


Figure 11: Efficiency of dependency patterns Mining

seconds. Based on the domain knowledge, Table 3 lists two critical event types (after event extraction) and the corresponding example messages. Both of the events would cause immediate system failures once they occur. Temporal pattern mining is leveraged to discover all the candidate events that have strong temporal correlations with 95%+ confidence. All the temporal patterns are discovered with time lags which characterize the temporal dependency between two events. As shown in Table 4, among all the discovered dependency patterns related to the two critical events, six of them (three for each critical event) are identified by domain experts from Huawei as the most possible causes.

In addition, as a deployed event log analysis system, system administrator requires a higher efficiency for responding quickly to the system alerts and problems. In this section, to assess the efficiency, we evaluate t-pattern mining under different scales of data extracted from Huawei log database. The empirical efficiency is evaluated by the CPU running time. As shown in Figure 11, each blue dot represents one sub-dataset, and the dotted line shows the trend of time consumption which illustrates that FLAP achieves nearly linear performance.

Table 4: Possible cause events discovered.

B (Critical Event)	**A** (Root Cause)	$A \xrightarrow[(\tau-\delta,\tau+\delta)]{} B$ (seconds)	Description
16	12	[1741, 1747]	Main LSP of Tunnel switches to backup LSP in HSB.
	51	[116, 122]	In the last 60 seconds, the number of times that the trap of mplsxcdown is suppressed.
	87	[118, 124]	The interface Gigabit Ethernet changed the baud from xxx bit to zzz bit.
132	6	[55, 61]	BFD session changed to Down.
	131	[658, 664]	Received GR end event.
	81	[0, 6]	Received and processed interface DOWN state.

To sum it up, dependency patterns can be discovered without specifying any time window size. It is a big challenge for traditional temporal pattern mining algorithm to determine the proper time window size. A small window size would causes the pattern mining algorithm to miss temporal patterns with large time lags, while a large window size results in large number of false positives.

Task 3: Failure prediction. The task of prediction is to identify whether there is system failures in the next hour, using the event log available for the past 1 hours. To build the predictive model, the historic logs are segmented into multiple event sequences with sliding windows, where each event chunk has a time range of 10 minutes. The segmented sequences are then divided into a training set and a validation set. For the weight cost, we set the positive samples 100 times the weight of the negative samples. For every 30 minutes, the events will be aggregated into one event chunk and the features will be extracted. Also, the models will be used to quantify the probability of the failure occurrence for the future 10 minutes. For the threshold, we set it as 20% by default.

Table 5: Failure prediction results on *Network X*.

Predicted \ Actual	Failure	No Failure
Failure	3	4
No Failure	1	85

Table 5 illustrates how FLAP predicts the future failures. We use the log generated in the following day as the test dataset. As shown in Table 5, FLAP is able to identify most of the failures of the system although it generates some false positive alerts. Overall, 75% of the failures can be identified ahead of time, which can significantly reduce the system downtime if all the alerts are seriously considered.

6.2 Evaluation Practice

Since its deployment, FLAP has been successfully used in many company activities, and brought several benefits in increasing the effectiveness of event log analysis: (1) The event dashboard presents a high level summary about the daily generated events of the monitored system. The system administrators are able to get a main idea of the daily behavior of the system without delving into the log details. Now the routine network inspection time is reduced from one hour to a few minutes. (2) Through event summarization and dynamic query form, the novice system administrators can quickly gain the insights of the monitored systems. Utilizing these modules, they can get the high level perspective of the generated events in terms of temporal dynamics, important event correlations, and critical event types. The average learning curve of the new system administrators has been shortened. (3) Using temporal pattern mining and failure prediction, system administrators effectively identify the problems of the monitored network and pinpoint the possible root causes, which reduces the downtime of the network.

7 RELATED WORK

An increasing number of event mining techniques have been successfully leveraged to analyze the system and optimize the management, especially in large-scale computing systems [12]. The data-driven techniques in system management can be broadly categorized into the following types: (1) pattern-based methods which aim to discover the hidden patterns, unexpected trends or other subtle relationships among events [16, 17, 19]; (2) temporal-based methods which discover temporal characteristics (e.g., time lags, evolving trends) and predict system behaviors [5, 23, 28]; and (3) summarization-based methods which summarize the characteristics (mainly including temporal dynamics) of the events with the given system logs [11, 20, 24, 25].

Compared with these previous studies, the temporal dependency pattern mining algorithm in FLAP is domain specific and meaningful for system management as these patterns can be well visualized

and be easily interpreted. Complex temporal dependencies can be constructed on the dependency patterns. In addition, an EM-based approach is proposed to discover the maximal likelihood model of time lags for the temporal dependencies in FLAP. Finally, the summarization approach used in FLAP uses inter-arrival histograms to capture the temporal relationships among same-type and different-type events and finds a collection of disjoint histograms to summarize the input event sequence. To the best of our knowledge, FLAP is the first integrated system to facilitate the data analytics for system event logs and support effective system management comprehensively.

8 CONCLUSION

In this paper, we developed FLAP, a comprehensive system aiming to provide the system administrators with an end-to-end solution for event log analysis. To facilitate event analysis, several state-of-the-art technologies in the areas of data mining, machine learning, and information retrieval have been integrated in the system. FLAP enables prompt data analysis and effective knowledge discovery in event logs. The initial success demonstrates that a practical data-driven solution considering system comprehensiveness, algorithm extensiveness, and user friendliness is able to facilitate administrators to better understand the system status and is informative to help increase the system availability.

ACKNOWLEDGMENTS

The work was supported in part by the National Science Foundation under Grant Nos. IIS-1213026, CNS-1126619, and CNS-1461926, Chinese National Natural Science Foundation under grant 91646116, Ministry of Education/China Mobile joint research grant under Project No.5-10, and an FIU Dissertation Year Fellowship

REFERENCES

[1] Amazon CloudWatch. http://aws.amazon.com/cloudwatch/.
[2] Scribe. https://github.com/facebookarchive/scribe.
[3] S.-H. Cha and S. N. Srihari. On measuring the distance between histograms. *Pattern Recognition*, 35(6):1355–1370, 2002.
[4] O. Etzion and P. Niblett. *Event processing in action*. Manning Publications Co., 2010.
[5] Z. Ge, J. Yates, L. Breslau, D. Pei, H. Yan, and D. Massey. Grca: A generic root cause analysis platform for service quality management in large isp networks. In *ACM Conference on Emerging Networking Experiments and Technologies*, 2010.
[6] P. D. Grünwald. *The minimum description length principle*. MIT press, 2007.
[7] HP. HP Operations Analytics: a New Analytics Platform to Support the Transformation of IT. *HP White Paper*, 2013.
[8] IBM. Monitoring the ibm http server on z/os from the tivoli enterprise portal. *IBM White Paper*, 2013.
[9] Y. Jiang, C. Perng, and T. Li. META: multi-resolution framework for event summarization. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 605–613, 2014.
[10] Y. Jiang, C.-S. Perng, and T. Li. Natural event summarization. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 765–774. ACM, 2011.
[11] J. Kiernan and E. Terzi. Constructing comprehensive summaries of large event sequences. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(4):21, 2009.
[12] T. Li. *Event Mining: Algorithms and Applications*, volume 38. CRC Press, 2015.
[13] T. Li, C. Zeng, Y. Jiang, W. Zhou, L. Tang, Z. Liu, and Y. Huang. Data-driven Techniques in Computing System Management. *ACM Computing Surveys*, 2017.
[14] T. Li and S. Ma. Mining temporal patterns without predefined time windows. In *IEEE ICDM 2004*, pages 451–454, 2004.
[15] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo. Failure prediction in ibm bluegene/l event logs. In *IEEE ICDM 2007*, pages 583–588, 2007.
[16] J.-G. Lou, Q. Fu, .Y. Wang, and J. Li. Mining dependency in distributed systems through unstructured logs analysis. *ACM SIGOPS Operating Systems Review*, 44(1):91–96, 2010.
[17] S. Ma and J. L. Hellerstein. Mining partially periodic event patterns with unknown periods. In *IEEE ICDE 2001*, pages 205–214. IEEE, 2001.
[18] M. L. Massie, B. N. Chun, and D. E. Culler. The Ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 30(7):817–840, 2004.
[19] K. Nagaraj, C. Killian, and J. Neville. Structured comparative analysis of systems logs to diagnose performance problems. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 353–366, 2012.
[20] S. Schneider, I. Beschastnikh, S. Chernyak, M. D. Ernst, and Y. Brun. Synoptic: Summarizing system logs with refinement. In *SLAML*, 2010.
[21] L. Tang and T. Li. Logtree: A framework for generating system events from raw textual logs. In *IEEE ICDM 2010*, pages 491–500, 2010.
[22] L. Tang, T. Li, Y. Jiang, and Z. Chen. Dynamic query forms for database queries. *IEEE Transactions on Knowledge and Data Engineering*, 2014.
[23] L. Tang, T. Li, and L. Shwartz. Discovering lag intervals for temporal dependencies. In *ACM SIGKDD*, pages 633–641, 2012.
[24] N. Tatti and J. Vreeken. The long and the short of it: summarising event sequences with serial episodes. In *ACM SIGKDD*, pages 462–470, 2012.
[25] P. Wang, H. Wang, M. Liu, and W. Wang. An algorithmic approach to event summarization. In *ACM SIGMOD*, pages 183–194, 2010.
[26] E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 407–418. ACM, 2006.
[27] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan. Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 117–132. ACM, 2009.
[28] C. Zeng, L. Tang, W. Zhou, T. Li, L. Shwartz, and G. Y. Grabarnik. An Integrated framework for Mining Temporal Logs from Fluctuating Events. *IEEE Transactions on Services Computing*, 2017.