IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 23 November 2022, accepted 19 December 2022, date of publication 21 December 2022,
date of current version 29 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3231454

SURVEY

A Survey of Log-Correlation Tools for Failure Diagnosis and Prediction in Cluster Systems

EDWARD CHUAH[1], ARSHAD JHUMKA[2], MIROSLAW MALEK[3], (Life Senior Member, IEEE), AND NEERAJ SURI[4], (Senior Member, IEEE)

[1]Computing Science Department, University of Aberdeen, AB24 3FX Aberdeen, U.K.
[2]Department of Computer Science, University of Warwick, CV4 7AL Coventry, U.K.
[3]Faculty of Informatics, University of Lugano, 6904 Lugano, Switzerland
[4]School of Computing and Communications, Lancaster University, Bailrigg, LA1 4YW Lancaster, U.K.

Corresponding author: Edward Chuah (thuan.chuah@abdn.ac.uk)

This work was supported in part by the Auspices of the European Union (EU) Horizon 2020 Research and Innovation Program [Cyber security competence for Research and innovation (CONCORDIA)] under Agreement 830927, and in part by the Security Lancaster under the Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/V026763/1.

ABSTRACT System logs are the first source of information available to system designers to analyze and troubleshoot their cluster systems. For example, High-Performance Computing (HPC) systems generate a large volume of heterogeneous data from multiple sub-systems, so the idea of using a single source of data to achieve a given goal, such as identification of failures, is losing its validity. System log-analysis tools assist system designers gain understanding into a large volume of system logs. They enable system designers to perform various analyses (e.g., diagnosing node failures or predicting node failures). Current system log-analysis tools vary significantly in their function and design. We conduct a systematic review of literature on system log-analysis tools and select 46 representative articles out of 3,758 initial articles. To the best of our knowledge, there is no work that studied the characteristics of log-correlation tools (LogCTs) with respect to four quality attributes including (a) spurious correlations, (b) correlation threshold settings, (c) outliers in the data and (d) missing data. In this paper, we (a) propose a quality model to evaluate LogCTs and (b) use this quality model to evaluate and recommend current LogCTs. Through our review, we (a) identify papers on LogCTs, (b) build a quality model consisting of the four quality attributes and (c) discuss several open challenges for future research. Our study highlights the advantages and limitations of existing LogCTs and identifies research opportunities that could facilitate better failure handling in large cluster systems.

INDEX TERMS System log-analysis, log-correlation tools, systematic literature review, quality model, failure diagnosis, failure prediction, cluster systems.

I. INTRODUCTION

System logs provide a wealth of valuable information that can help system designers understand the behaviour of large cluster systems. Syslog is the standard logging protocol used by large cluster systems to record events generated by system software or hardware devices [1]. Syslog was conceptualized over four decades ago as an event record store for software applications or devices that allowed computer system designers to analyze the generated data. Using the Syslog protocol standard, standardized logging allowed system designers to consolidate logging data from a wide variety of devices and software applications over the network. While Syslog is currently extensively used by many devices (routers, printers, filesystems, etc.) and various implementations of the standard system logging protocol exist for several operating systems, more recently, the changes in usage patterns and increased data volumes are highlighting its limitations. For example, HPC systems allow users to run complex simulation jobs and machine learning algorithms, and parallel applications have become the main consumer of resources on these systems. These systems

The associate editor coordinating the review of this manuscript and approving it for publication was Dongxiao Yu.

generate a large volume of data and different types of data, which makes it difficult to reduce the volume of data to process [2].

To address this issue, a wide variety of log-analysis tools have been developed for reducing the amount of system logs. These tools implement different filtering techniques, statistical techniques, data mining methods or machine learning algorithms. They are also designed for different goals, e.g, diagnosing node failures [3], predicting node failures [4], detecting anomalies [5] and analyzing system resource use [6]. However, there is a gap between the needs of system designers and researchers. System designers are not aware of the quality attributes of LogCTs. System designers do not have time to search through the large body of literature to identify a LogCT which can meet their needs and they do not have the time or the resources to review the characteristics of each LogCT. Furthermore, researchers focus on developing new log-analysis techniques whereas system designers are also interested in comparing LogCTs based on other useful aspects.

To bridge this gap, our paper makes the following contributions:

- We conduct a systematic review of literature in state-of-the-art LogCTs.
- We identify and integrate quality attributes of LogCTs. We identify quality attributes of LogCTs through our systematic literature review, use these attributes to build a quality model and apply our quality model to compare several LogCTs.
- We discuss the advantages and limitations of current LogCTs, and recommend LogCTs for use cases.
- We also present directions for future research.

We conduct our literature review using guidelines given by Khan et al. [7]. Their approach for conducting a systematic literature review consists of five steps: (a) frame the research question, (b) identify the relevant work, (c) assess the quality of the studies, (d) summarize the evidence and (e) interpret the findings. We use this literature review approach and define our research questions, identify the relevant LogCT articles, evaluate the quality of the LogCTs, collate the LogCT articles and discuss our results. We search for articles on system log-analysis in multiple databases, including but not limited to the digital libraries of IEEE, ACM, Springer and Elsevier, and obtained an initial set of 3,758 articles. We identify a set of seed papers and perform forward snowballing to obtain all the articles that cite those seed papers, and backwards snowballing to obtain all the references in those seed papers. We apply a list of general criteria, inclusion criteria, exclusion criteria and quality assessment criteria and obtained 46 representative articles, from which 14 unique LogCTs are identified.

The effectiveness in diagnosing the cause of a system failure or predicting a system failure rely upon the LogCT ability to detect spurious correlations, determine the optimal threshold settings, identify outliers in the data and handle missing values in the data. A spurious correlation is a mathematical relationship in which two or more log-messages are associated but are not causally related. The threshold settings remove redundant log-messages which can reduce the amount of log-messages for analysis. Outliers in the data can reduce or increase the value of the correlation coefficient which has an impact on the standard measure of correlation. Missing values in the data can decrease a statistical test's ability to identify relationships between log-messages. If a LogCT did not take into account spurious correlations, threshold settings, outliers in the data and missing values in the data, it can produce a wrong diagnosis or a wrong prediction of a system failure. Thus, LogCTs must detect spurious correlations, determine the optimal threshold settings, identify outliers in the data and handle missing values in the data.

We compared these LogCTs and showed that (a) there is not one LogCT that can meet all the four quality attributes of spurious correlation, threshold settings, outliers in the data and missing data, (b) researchers have been working to improve failure diagnosis or failure prediction accuracy by implementing different approaches using multiple methods from statistics and data mining, but few have considered spurious correlations which can lead to a wrong diagnosis or false prediction and needs to be investigated so that the risk of making a wrong diagnosis or false prediction is reduced, (c) selecting the optimal correlation threshold settings is challenging and requires significant manual effort due to the different range of values used by each approach, so studying how these parameters can be tuned to improve diagnostics or prediction accuracy is important, and (d) few LogCTs have investigated the impact of outliers or missing log-events on the diagnostics or prediction accuracy, so designing novel correlation approaches which are robust against outliers and missing log-events is important.

Recent survey papers have provided valuable knowledge on state-of-the-art log-analysis tools [8], failure diagnosis approaches [9] and failure prediction methods [10]. He et al. [8] reviewed several automated event-log processing methods, event-log processing tools and open-sourced datasets and showed the value of these methods, tools and datasets for reliability engineering. Kavulya et al. [9] reviewed several automated failure diagnosis approaches. Salfner et al. [10] developed a taxonomy that captured the wide range of online failure prediction techniques and explained the different techniques in detail. Differently to He et al. [8], Kavulya et al. [9], and Salfner et al. [10], we review current LogCTs and propose a quality model for evaluating state-of-the-art LogCTs. El-Masri et al. [11] reviewed automatic log-abstraction techniques and built a quality model consisting of seven quality attributes. Their quality attributes are mode, coverage, independence of delimiters, efficiency, scalability, independence of system knowledge and effort in tuning parameters. They applied their quality model and evaluated 17 log-abstraction tools. In contrast, our quality model contains four different quality attributes. The quality attributes are spurious correlations,

TABLE 1. Distinct functions of LogCTs. "MI" means there is no information on this function.

Tool	Preprocessing	Support multiple types of logs	Automated	Customizable	Goal
SEC [12]	Yes	Yes	MI	Yes	Monitoring
LogMaster [13]	Yes	Yes	Yes	MI	Failure prediction
A. Gainaru *et al.* [4]	Yes	Yes	Yes	MI	Failure prediction
Z. Lan *et al.* [14]	Yes	Yes	Yes	MI	Failure prediction
RPTCN [15]	Yes	MI	Yes	MI	Resource prediction
X. Fu *et al.* [16]	Yes	Yes	Semi-automated	MI	Failure diagnosis & prediction
SIG [3]	Yes	Yes	Yes	MI	Failure diagnosis
FDiag [17]	Yes	No	MI	Yes	Failure diagnosis
Baler [18]	Yes	Yes	Yes	MI	Failure diagnosis
Z. Zheng *et al.* [19]	Yes	Yes	Yes	MI	Failure diagnosis
UiLog [20]	Yes	Yes	Yes	MI	Fault diagnosis
CORRMEXT [21]	Yes	Yes	MI	MI	Error propagation
ANCOR [22]	Yes	Yes	MI	Yes	Failure diagnosis
IFADE [23]	Yes	Yes	MI	MI	Failure diagnosis

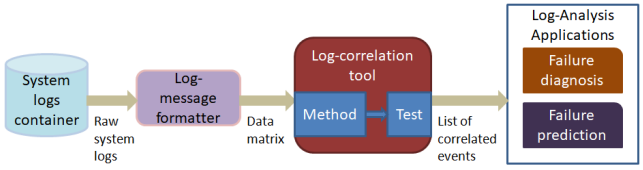

FIGURE 1. Log-correlation pipeline.

correlation threshold settings, outliers in the data and missing data. We apply our quality model to evaluate 14 LogCTs. The results presented in this paper are based on a thorough review of current LogCTs. We summarize their functions in Table 1 and describe the approach implemented by these LogCTs in Section V.

The remainder of this paper is organized as follows. We give a background on the log-correlation process in Section II. We motivate the use of LogCTs in Section III. Then, we describe the design of our study in Section IV. We organize and summarize the approaches implemented by the 14 LogCTs in Section V, identified through our systematic review of literature on LogCTs. We present our LogCT quality model in Section VI, evaluate the 14 LogCTs and present our results and observations in Section VII. We discuss the threats to the validity of our results in Section VIII and conclude with a summary and future work in Section IX.

II. LOG-CORRELATION PROCESS

A typical log-correlation process is shown in Figure 1. It consists of three phases: (a) log-message formatting, (b) a log-correlation tool and (c) log-analysis applications. Next, we describe each phase of the log-correlation process.

A. PHASE 1: LOG-MESSAGE FORMATTING

Most system log-messages contain a time stamp, source and message [1]. An example of a Rationalized message log [24] and a Syslog message [1] are shown in Figure 2 and Figure 3 respectively.

```
time:1293552419
host:i101-403
jobid:1743653
prog:kernel
0:<3>BUG: soft lockup detected on CPU#%d, pid:%d, uid:%u, comm:%s
1:14
2:5656
3:0
4:ldlm_bl_04
```

FIGURE 2. Rationalized log-message.

```
Jan 1 01:21:20 oss30 kernel: LustreError: 0:0:(ldlm_lockd.c:305:waiting_locks_callback())
### lock callback timer expired after 51s: evicting client at *.*.*.*@o2ib ns: filter-scratch-
OST0100_UUID lock: 00000102347cbb80/0x344dcb188d1dcc4a lrc: 3/0,0 mode: PW/PW
res: 30607838/0 rrc: 2 type: EXT [0->18446744073709551615] (req 0->4095) flags: 0x20
remote: 0x5275d77aa11d3abd expref: 4 pid: 7941 timeout 4438041253
```

FIGURE 3. Syslog message.

From Figure 2, we observed that the Rationalized log-message contains the following features and feature values: (a) time stamp (`time:1293552419`), (b) host id (`host:i101-403`), (c) job-id (`jobid:1743653`), (d) program (`prog:kernel`), (e) error message (`0:<3>BUG: soft lockup detected on CPU\ldots `) and (f) additional variables associated with this log-message (`1:14`, `2:5656`, `3:0`, `4:ldlm_bl_04`). From Figure 3, we observed that a sequence of feature values is contained in the Syslog message.

The Rationalized log-message and Syslog message contain features such as a time stamp, a host identifier, a program identifier and an error message string. However, the features in the Syslog message and Rationalized log-message are organized differently. In the Rationalized log-message, each feature and its value is organized in one row. In the Syslog message, the feature values are represented in a sequence where each feature value is separated from another feature value by a single space. There is also no job identifier in the Syslog message. The format of system logs generated on one cluster system may be different to the format of system logs generated on another cluster system, which makes it

difficult to parse the data because a different parser has to be developed for each type of system log. However, most system logs contain three basic features: (a) a time stamp, (b) a host identifier and (c) an error message string. Thus, the system logs can be converted into a standard format. For example, a standard formatted system log may contain the following features:

```
job id, month, day, hour, minute,
second, node id, program name, message
```

Most data analysis methods use a data matrix as its input [25], [26]. Once the raw system logs are converted into the standard formatted log, the logs are then converted into a data matrix. A data matrix is a two-dimensional structure where each column represents a feature, each row represents a sample, and each cell represents a count for a feature sample. The occurrences of *Constants* in the system log can be represented in a data matrix where each column represents a constant, each row represents a sample such as time or node or job, and each cell contains the count for a constant sample. A *Constant* is a sequence of English-only words contained in the message part of a system log.

B. PHASE 2: LOG-CORRELATION TOOLS

LogCTs reduce the number of events in the system logs on which further analysis can be applied. The objective of log-correlation is to identify relationships between log-events, to group log-events that are correlated, and to separate log-events that are uncorrelated. System designers may look through the list of correlated events to determine the likely cause of a system problem [27]. For example, if a system designer wants to determine whether a compute node soft lockup is caused by a crashed program, they can use the list of correlated events to see if it contains ''soft lockup'' and ''segmentation fault'' events. If the list of correlated events contains strongly positive correlated ''soft lockup'' and ''segmentation fault'' events, it shows that a crashed program is a likely cause of compute node soft lockups [21].

In general, LogCTs are composed of two components: (a) a method and (b) a test. The *method component* receives a data matrix and applies a correlation algorithm to produce correlation coefficients for all pairs of log-events. When the correlation coefficients are produced, the *test component* uses statistical significance tests to identify log-events which are uncorrelated and log-events which are strongly correlated.

Correlating log-messages requires LogCTs to address four challenges. These challenges are described as follows:

1) SPURIOUS CORRELATIONS

A spurious correlation is a measure in which two or more events are strongly correlated, but they are in fact not causally linked, usually because of the presence of a third or confounding event [28]. System designers must manually infer a likely cause of a system problem from the list of correlated events, which is difficult because some events can be strongly correlated purely by chance without one event actually causing the other event to occur. Such spurious correlations can lead to a wrong diagnosis or wrong prediction of a failure [27].

2) THRESHOLD VALUE

System designers must manually choose a threshold value and use this value to remove redundant log-events and identify the correlated log-events or clusters of log-events for further analysis, which is tedious because choosing the optimal threshold value in LogCTs is time consuming. Some system designers may decide to use a small range of values and determine a threshold value when the same number of correlated events or well-separated clusters of log-events are obtained.

3) OUTLIERS IN THE DATA

An outlier in the dataset is the value of a data point that falls outside the range of values of the remaining data points. Depending on where the outlier lies in the dataset, the outlier may increase or decrease the value of the correlation coefficient in relation to the other data points, which is challenging because it can have an impact on the standard measure of correlation [29] or influence the final cluster configuration [30]. Some LogCTs have addressed this issue by using alternative robust measures such as rank-correlation [31].

4) MISSING DATA

Missing values in a dataset occur when no data value is stored for that event. Missing data causes two problems [32]: (a) a decrease in a statistical test ability to identify relationships in the data and (b) it may bias correlation coefficients downwards.

C. PHASE 3: LOG-ANALYSIS APPLICATIONS AND DATASETS

Log-analysis is used by many applications e.g., detecting web attacks [33], characterizing usability of interactive applications [34], diagnosing system problems [3] and predicting node failures [31]. Next, we describe some of these applications, their influence on log-correlation tools and publicly available datasets.

1) FAILURE DIAGNOSIS

Failure diagnosis is an approach for tracing a fault through the identification of its symptoms [35]. Failure diagnostics tools typically use correlation methods such as Pearson correlation [3], [19], [36] to analyze system failures. Thus, LogCTs are a prerequisite for failure diagnostics tools to provide the list of correlated events needed for inferring the likely cause of a system problem.

2) FAILURE PREDICTION

Failure prediction is an approach for determining the probability that a system will fail [37]. In general, failure prediction tools select a subset of features for training

a prediction model by using feature selection techniques such as correlation-based feature selection. Thus, LogCTs are a prerequisite for failure prediction tools to generate a prediction model needed to determine the probability that a failure event will occur accurately.

3) MODEL TRAINING

Correlation-based Feature Selection (CFS) uses a correlation measure and a heuristic search strategy to select a subset of the relevant features [38]. These features are then used to train data mining models to predict node failures. For example, Di et al. [39] used a correlation algorithm to remove highly correlated log-events prior to training their modified K-Means clustering technique. Alharti et al. [40] also removed the highly correlated log-events prior to training their sentiment analysis model. Therefore, LogCTs are a prerequisite for model training to (a) shorten the training time and (b) increase the prediction power of the prediction model by selecting only the relevant features for training the model [39], [40].

4) PUBLICLY AVAILABLE SYSTEM-LOGS DATASETS

To facilitate research in system log-analysis, failure logs, accounting logs, lists of software libraries used by HPC jobs and resource usage data are available on the public domain. The Computer Failure Data Repository (CFDR) contain nine years worth of failure data that were collected on HPC systems operated by Los Alamos National Laboratory [41]. The FRESCO job failure and performance data repository contain jobs performance data and accounting logs that were collected on the Conte and Stampede-1 supercomputers, as well as software libraries usage data, downtime, scheduled and unscheduled outage logs that were collected on Conte [42].

III. MOTIVATION

Data centers operate several large cluster systems. These cluster systems generate a large volume of system logs that contain valuable information for diagnosing system problems or predicting system failures. Off-the-shelf log-analysis frameworks (e.g., Splunk, SolarWinds, Graylog or Stackify) provide data integration, real-time monitoring and workflow development pipelines. However, these log-analysis frameworks provide only basic analytics and thus, system designers have to develop their own tools to search through vast amount of system logs in order to identify the sequence of events which lead to a system failure [43]. Furthermore, these cluster systems generate different types of system logs such as resource use data, accounting logs and scheduler logs and millions of log-messages are recorded each day. Therefore, system designers require LogCTs that can offer advanced diagnostics or prediction of system failures.

There are many LogCTs presented in the literature. Thus, system designers have at their disposal a wide variety of LogCTs to choose from. However, they need to select a LogCT with quality attributes that best suits their particular use case. For example, in failure diagnosis, a LogCT must not identify correlated events which do not represent a causal relationship [27]. The LogCT should also be robust against noise or missing log-events. Furthermore, the correlation threshold settings in LogCTs are used to create a fault model or a prediction model. Therefore, LogCTs must enable system designers to tune the threshold settings so that an accurate diagnosis or prediction can be obtained [44].

IV. STUDY DESIGN

We studied the guidelines for conducting a literature review [7]. Based on these guidelines, we review current LogCTs as follows: (a) frame the research question, (b) identify related works, (c) assess the quality of papers, (d) summarize the methodology and (e) compare similarities and differences between LogCTs.

A. RESEARCH QUESTIONS

To review state-of-the-art LogCTs along with their challenges, we use answers to the following research questions to propose a quality model. Then, we use the quality model to evaluate current LogCTs. The research questions are given as follows:

- Q1 What are the state-of-the-art log-correlation tools?
- Q2 What are the quality attributes of these log-correlation tools?

B. SEARCH APPROACH

We studied papers that are published in journals and conferences. These papers are written in English and published between January 1993[1] and June 2022. We conducted our literature search using Google Scholar which provides access to multiple digital libraries, which include the IEEE, ACM, Springer and Elsevier among many others. We combined seed papers from Google Scholar and papers snowballed from Scopus. We used the following queries to search for the seed papers: "`log correlation`" or "`log analysis`" or "`failure prediction`" or "`failure diagnosis`".

C. SELECTION APPROACH

We screened the articles using three screening stages (a) general criteria (type of paper, publication time frame and subject area), (b) criteria for inclusion and exclusion and (c) quality of the paper. Our *general criteria* are given as follows:

- The article shall be published between January 1993 and June 2022.
- The article shall be published in a conference or journal.
- The article shall pertain to system log-analysis tools for cluster systems and cloud computing systems. These systems generate a large volume of heterogeneous data

[1]We decided to start at 1993 because SWATCH is one of the first system logs monitoring and analysis tool [45].

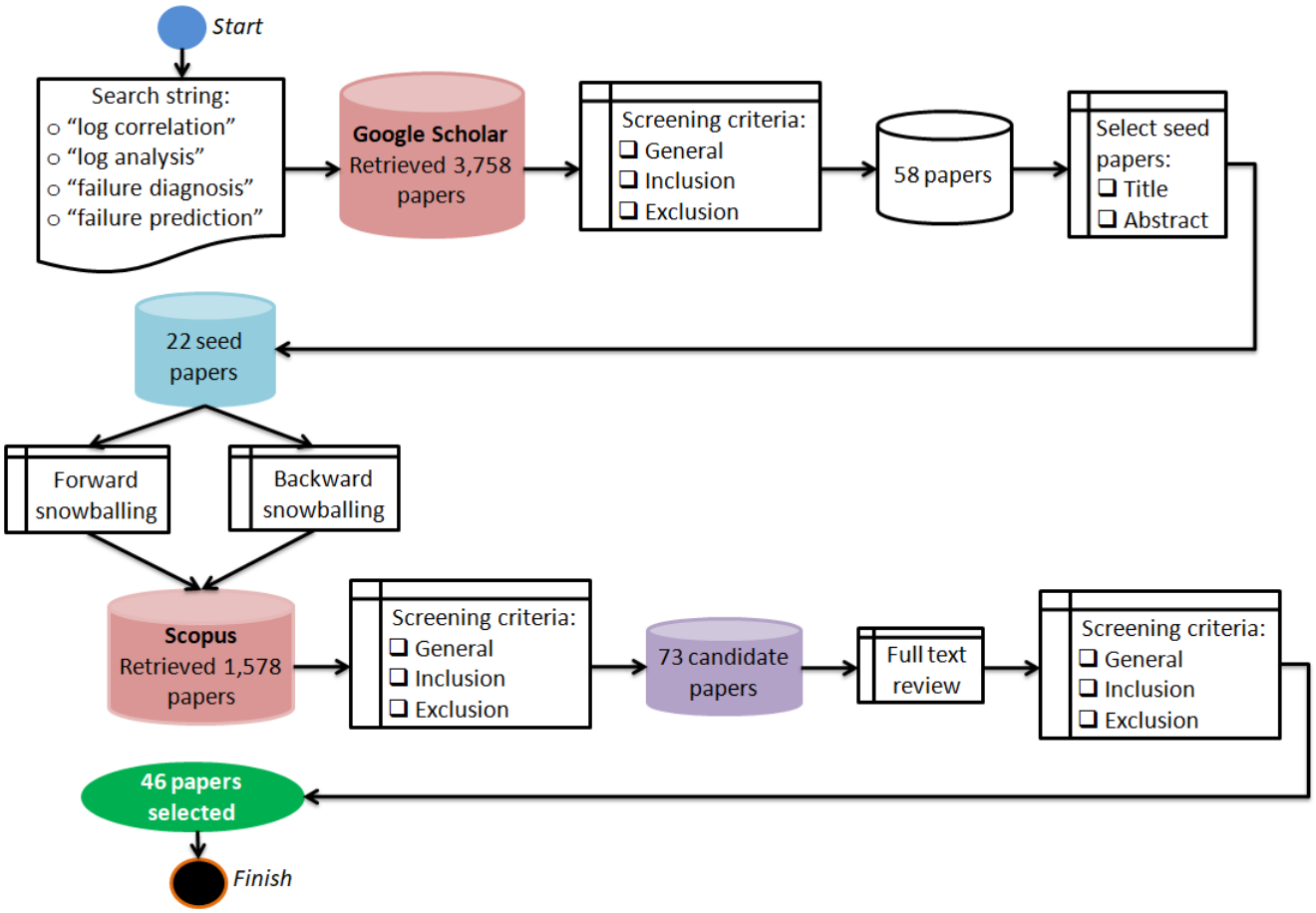

FIGURE 4. Search and selection workflow.

from multiple subsystems and system log-analysis tools enable system designers to perform various analyses such as diagnosing node failures or predicting node failures. Thus, we focused on articles on system log-analysis tools for cluster systems and cloud computing systems.

Our *inclusion criteria* are given as follows:

- The article shall be written in English.
- The article shall present, propose, develop or implement a log-correlation technique.

Our *exclusion criteria* are given as follows:

- When two or more articles present, propose, develop or implement the same or a similar script, statistical technique, data mining method or machine learning algorithm, at least one article shall be excluded from the final set of articles.
- Articles that are not peer-reviewed.
- Articles that describe how to write system logs.
- Articles that require access to source code.
- Articles that present message logging pipelines or message logging architectures.

Our *quality assessment criteria* are given as follows:

- Is the research problem clearly described in the article?
- Is the proposed method clearly described in the article?
- Does the article describe how the proposed method is evaluated?
- Does the proposed method address the research problem set by the researchers?

Our search and selection workflow is shown in Figure 4. It consists of three steps (a) identify seed articles, (b) identify candidate articles and (c) identify the final set of articles.

1) SEED ARTICLES

We conducted our search by executing our search queries on Google Scholar. We retrieved 3,758 articles. Then, we filtered these articles using our general criteria, inclusion and exclusion criteria and retained 58 articles. We reviewed the title and abstract in these articles and divide them into two groups: (a) include the article or (b) exclude the article. We obtained 22 seed articles.

2) CANDIDATE ARTICLES

We searched in Scopus for (a) all articles that cite the 22 seed articles (forward snowballing) and (b) all references in the seed articles (backward snowballing). We obtained 1,578 articles. Then, we filtered these articles using our general criteria, inclusion and exclusion criteria and retained

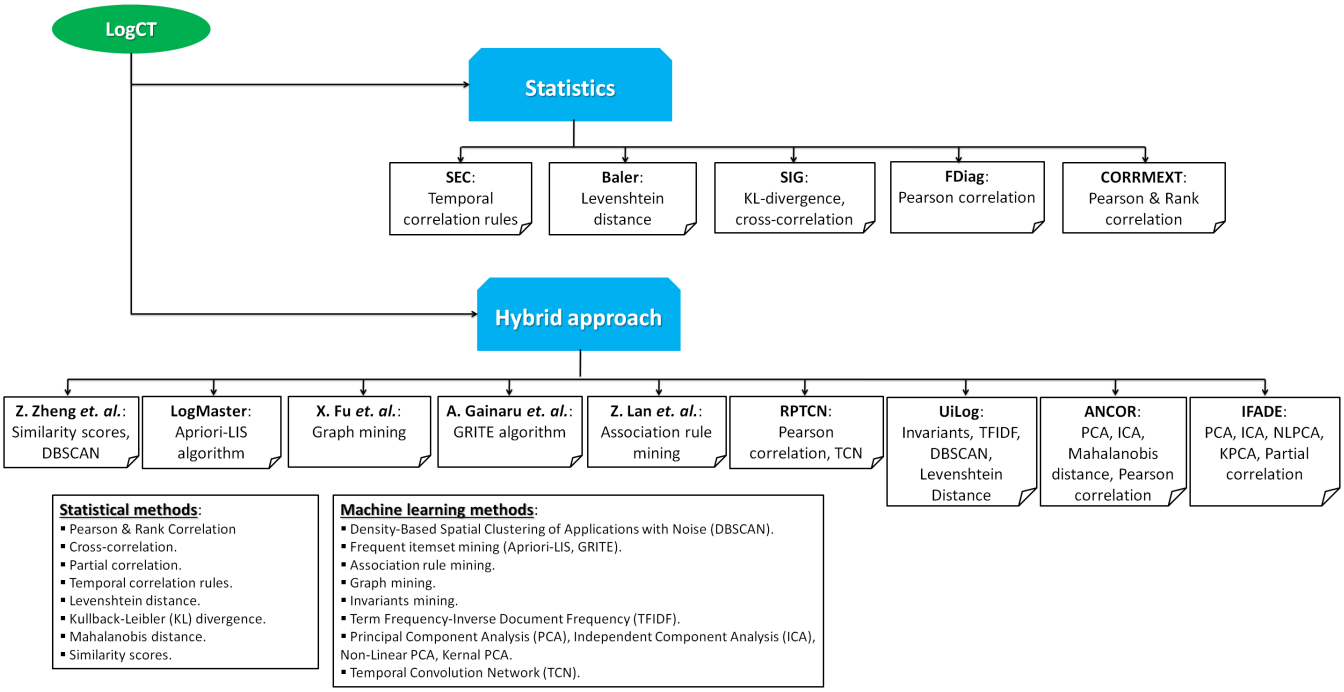

FIGURE 5. Categories of log-correlation tools.

51 articles. We combined the 51 articles and 22 seed articles into a set of 73 candidate papers.

3) SELECTED PAPERS

Each candidate article is reviewed by two authors independently. We read the details in all 73 candidate papers and evaluated each article according to our inclusion criteria, exclusion criteria and quality assessment criteria. Both authors compared the data and resolved disagreements by consensus. Then, we collated the decisions from the two authors and obtained the final set of 46 articles.

D. DATA EXTRACTION

We reviewed all 46 articles in detail and obtained data about the following:

- Methods, algorithms and approaches in state-of-the-art LogCTs.
- Desired features of a LogCT, their definitions and their classification criteria.

We collated and compared the data obtained on the characteristics of LogCTs into several quality attributes. We use these quality attributes to build our quality model, and obtain the results and evaluations of LogCTs with respect to these quality attributes.

E. DATA SYNTHESIS

We combined and summarized the extracted data. The catalogue of state-of-the-art LogCTs is given in Section V. Their evaluations with respect to the identified quality attributes is given in Section VII.

F. STUDY SCOPE

Our study considered open-source LogCTs and LogCTs where their source code is not available due to the distribution policy of the data center. We did not consider log-correlation tools for security nor check the source code or implement the approach to assess its correctness as they would require access to all the source code and substantial resources beyond the scope of this paper.

V. LOG-CORRELATION TOOLS

In this section, we describe the LogCTs identified from our systematic literature review. We organize these LogCTs based on the category of algorithms implemented by the LogCT. The categories of algorithms and their LogCTs are shown in Figure 5.

A. APPROACHES THAT USE STATISTICAL METHODS

The following LogCTs developed approaches that use statistical methods and analyzed large quantities of unstructured system logs generated on cluster systems. Next, we describe the approach implemented by these LogCTs.

1) SIMPLE EVENT CORRELATOR

The Simple Event Correlator (SEC) is a log-analysis tool that analyzes log-events with respect to the time log-events are received [12], [46]. It uses static rules and provides explicit matching of events, counting operations and aggregation of events. SEC receives input events from a file stream and produces output events by executing a user-specified shell command. The input events are comprised of regular expressions and rules that are contained and stored in

configuration files. Nine different types of rules are supported by SEC. The rules range from matching a pair of events to matching multiple events within a given time-window or threshold. The list of actions includes log a message, write a line into a file, execute an external shell script or program and generate a new input event which can be matched by other rules. To handle missing log-events, SEC can generate log-events periodically to check for a missing log-event.

2) MINING CLUSTERS OF LOG-MESSAGES FOR FAILURE DIAGNOSIS

Baler is a system logs-mining tool that extracts message patterns from a large volume of unstructured system logs [18]. Differently to SEC, Baler did not use rules to match log-messages. Instead, Baler implements a novel log-message clustering engine. Their clustering engine consists of two steps (a) parsing log-messages and (b) clustering log-messages. In the first step, the system logs are parsed by a regular expression that uses alpha-numeric words, white space, colon, bracket and other symbols. In the second step, the Levenshtein distance measure is used to assign log-messages into clusters where each cluster contains log-messages that are similar. The Levenshtein distance measure calculates the similarity between two log-messages by counting the number of insert, delete or replace operations that are required to change one log-message to the other. To determine if two log-messages are similar, the authors specified two criteria. The first criterion specify that two log-messages are similar if the number of insert, delete or replace operations is less than a given threshold. The second criterion specifies that two log-messages are similar if the ratio of change counts and message length is less than a given threshold.

3) STRUCTURE-OF-INFLUENCE GRAPHS

Oliner et al. [3] proposed *Structure-of-Influence Graphs (SIGs)* to diagnose problems in complex production systems. Their process consists of four steps for constructing a SIG: (a) decide what information is to be used from each component, (b) use anomaly signals to measure system behaviour, (c) compute the strength of the pairwise correlation between anomaly signals and (d) build a graph where system components are represented as nodes and the strength and delay between system components are represented by edges. In the first step, information produced by system components is used to develop two models. The first model uses message timings and the second model uses information content of the message terms. In the second step, anomaly signals are computed using the Kullback-Leibler (KL) divergence, which is a statistical technique that measures how one probability distribution is different from another probability distribution. In the third step, the Pearson correlation algorithm is used to obtain a correlation coefficient for two anomaly signals. In the fourth step, a SIG is constructed for a system with n components. However, constructing a SIG for n components requires a quadratic time complexity. To address this problem, a subset of the SIG is constructed. They evaluated their method under the influence of measurement noise, message loss and tainted training data, and showed that SIGs are robust against uniform message loss, degrade gracefully when timing measurements contain noise and that its ability to detect influence did not depend on clean training data. Oliner and Aiken [47] enhanced SIGs with Principal Component Analysis (PCA) and performed online detection of interactions between system components of large supercomputers.

4) DIAGNOSING LUSTRE FILE-SYSTEM FAILURES

FDiag is a log-analysis tool that filters streams of interleaved log-events to identify events that are relevant to the diagnosis of a given failure [17]. Differently to SIGs, FDiag did not correlate anomaly signals. Instead, FDiag identifies message templates which are strongly positive correlated. FDiag consists of three components (a) a message template extractor, (b) a statistical event correlator and (c) an episode constructor. The message template extractor extracts message templates from unstructured system logs and produces a standard data format which adds structure to the unstructured logs. A message template contains a sequence of *Constants*, which consists of a sequence of English-only words. Constants are extracted by specifying a regular expression and applying it to log-messages in the system logs. The statistical event correlator uses the Pearson correlation algorithm to identify strongly positive correlated message templates for a given symptom event, and a correlation threshold value of 0.75 is used. If the correlation strength between two log-messages range between 0.75 and 1, these log-messages are extracted and stored in a list of strongly positive correlated log-messages. The episode constructor uses two different heuristics to construct sequences of correlated log-messages which lead to a node failure. FDiagV3 enhanced FDiag with PCA and Independent Component Analysis (ICA) feature extraction algorithms and uncovered five previously unknown causes of node failures [48].

5) IDENTIFYING ERROR PROPAGATION AND RECOVERY PATTERNS IN CLUSTER SYSTEMS

CORRMEXT (*COR*relating *R*esource use and *ME*ssage logs and e*X*tracting *T*imes) is a new diagnostics framework that analyzes patterns of system errors and generates reports on the success and failure of error recovery protocols [21]. FDiag, ANCOR (see Section V-B8) and CORRMEXT assume that when two message types are strongly positive correlated, these correlated messages can be used to determine the likely cause of a given system failure. Similar to FDiag and ANCOR, CORRMEXT identifies correlations of message types. But differently to FDiag and ANCOR, CORRMEXT identifies the earliest time of occurrence of a system problem. While FDiag and ANCOR focused on failure diagnosis, CORRMEXT focused on identifying error propagation patterns. It processes Rationalized message logs [24], Syslogs [1], and resource usage data [49].

CORRMEXT consists of three modules (a) a data types extractor module, (b) a correlation module and (c) a time-bins extraction module. The data types extractor module is composed of two data type extractors. A resource use extractor extracts resource use counters from resource usage data and organizes counters of resource use counters by time-bins into a resource use data matrix. The authors defined a time-bin as a time window of one fixed time interval. A message types extractor extracts message types from Rationalized message logs and Syslogs and organizes counts of message types by time-bins into a message types data matrix. The correlation module uses Pearson correlation and Spearman-Rank correlation algorithms to obtain correlation coefficients for the resource use counters and correlation coefficients for the message types. It uses a correlation threshold value of 0.8. If the correlation strength of two message types or two resource use counters is greater than or equals to 0.8, then the message types or resource use counters are stored in a list of strongly positive correlated message types or strongly positive correlated resource use counters. It uses Fisher's z-transform to test the significance of all correlation coefficients and Bonferroni Correction to identify false positives. A fixed threshold value is used to identify strongly positive correlated resource use counters and strongly positive correlated message types. The time-bins extraction module obtains differences between the variance of two adjacent time-bins for the strongly correlated resource use counters, and separately for the strongly positive correlated message types. EXERMEST (*EX*tracting f*E*atures and co*R*relating resource use counters and *MES*sage *T*ypes) enhanced CORRMEXT with several feature extraction methods and linked significant resource use counters and message types with node failures and error recovery protocols [50].

B. HYBRID APPROACHES THAT INTEGRATE STATISTICAL METHODS AND MACHINE LEARNING ALGORITHMS

The following LogCTs developed approaches that integrate statistical methods and machine learning algorithms, and analyzed large volumes of system logs generated on cluster systems or cloud computing systems. Next, we describe the approach implemented by these LogCTs.

1) 3-DIMENSIONAL ROOT-CAUSE DIAGNOSIS

Zheng et al. [19] presented a novel root-cause diagnostics mechanism to diagnose failures in large cluster systems. Their diagnostics method identifies HPC application, system software and hardware failures. Their process for identifying the failure layer, time and location of a fatal event consists of four steps: (a) preprocess data, (b) integrate information across three types of logs, (c) identify the failure layer and (d) identify the location and time of events which caused a given failure. In the first step, wavelet transformation is used to remove noise in environmental logs, RAS (Reliability, Availability and Serviceability) logs and job logs. Then, temporal-spatial filtering is used to identify and remove redundant events in RAS logs. To group jobs that are contained in job logs, the Density-Based spatial Clustering of Applications with Noise (DBSCAN) algorithm is used to identify jobs that share similar characteristics automatically. In the second step, environmental features (e.g., power consumption, heat generated) closely related to fatal events are extracted. In the third step, a job similarity score is computed and used to identify jobs that were interrupted by a specific type of fatal event. Different threshold values were used to distinguish HPC application failures, hardware failures or system software failures. In the fourth step, a dynamic time window generation technique and probabilistic causality pruning technique is used to identify the location and time of the cause of a given fatal event.

2) MINING CORRELATIONS OF LOG-EVENTS

Fu et al. [13] proposed an approach called *LogMaster* to mine correlations of log-events. LogMaster consists of several innovative algorithms that perform log-preprocessing and filtering, mine event correlations, construct event correlation graphs and predict failure events. In the log-preprocessing and filtering algorithm, two types of events are filtered (a) repeated events and (b) periodic events. Repeated events are events that occur repeatedly in a short window of time. A predefined threshold is used to determine if the time interval between two events is less than the threshold. If the time interval falls below the threshold, the event is removed. Periodic events are events that occur periodically with a fixed time interval. A predefined threshold is used to determine if the count and percentage of all events in the same time interval is higher than the threshold. If the count and percentage are higher, then only one event is retained. The event correlation algorithm uses a modified Apriori-LIS algorithm. The modified algorithm apply an event filtering policy that specifies that only correlated events that occur on the same node, the same application or the same event type are analyzed. Event rules are represented in a directed acyclic graph (ECG). Each ECG contains correlated events for one node or correlated events for multiple nodes. The event prediction algorithm is based on event correlation graphs. It uses a prediction probability threshold to warn if the probability of predicted events exceeds this threshold.

3) DIGGING DEEPER INTO CLUSTER LOGS FOR FAILURE DIAGNOSIS AND PREDICTION

Fu et al. [16] proposed an approach to diagnose and predict failures from cluster system logs. Similar to LogMaster, their approach uses event correlation mining and event correlation graphs. Moreover, they analyze relationships between non-fatal events and fatal events and use fine grained information such as the type of failure and location of a node to provide a deeper and more accurate diagnosis. Their approach consists of three steps: (a) mine event causal dependency graphs (CDGs), (b) extract failure rules and (c) deduce new failure rules. In the first step, frequent event sequences are clustered to form correlated groups of events. In the second step, failure

rules are extracted from CDGs. They identify a failure rule by matching event IDs from the first precursor event to each successor event upto a given failure event. To determine candidate failure rules, the authors use two metrics to obtain a confidence value, which is then assigned to each failure rule. The metrics are *rule false positive* and *rule false negative*. If the confidence value of a candidate failure rule falls below a predefined threshold, the failure rule is removed. In the third step, failure rules extracted during the second step are used to identify failure event generating processes and new failure rules are deduced from these failure event generating processes.

4) A NOVEL METHOD FOR FAULTS CLASSIFICATION AND DIAGNOSIS

In [20], the authors proposed a novel classification method for fault logs and implemented a fault analysis and diagnosis system called *UiLog*. UiLog consists of three components: (a) a fault log-collection module, (b) a fault log-analysis module and (c) a fault log-correlation module. The fault log-collection module stores system and software event logs in a log-information database. The fault log-analysis module implements a new fault classification method, which preprocesses the logs, extracts invariants, filters message templates, generates a fault matrix and classifies faults. The fault log-correlation analysis module uses different time window sizes to handle truncation and collision errors caused by overlapping fault sequences.

5) FAULT PREDICTION UNDER THE MICROSCOPE

Gainaru et al. [4] presented an approach that merged signal processing techniques with the Event Log Signal Analyzer (ELSA) toolkit and provided a detailed analysis on their prediction method. Their approach and FDiag extract message templates from system message logs and apply correlation analysis to obtain correlated log-messages. However, their goals are different. FDiag focused on HPC system failure diagnosis while the approach in [4] focused on predicting HPC system failures. In [4], the authors described the details of their approach. It consists of four modules: (a) event log-preprocessing, (b) outlier detection, (c) signal correlation and (d) location correlation. In the event log-preprocessing module, the Hierarchical Event Log Organizer (HELO) is used to extract message templates from event logs. Message templates correspond to different system events and they are represented as regular expressions, which describe a set of syntactically related messages. In the outlier detection module, a data mining algorithm based on a moving time window is used to identify erroneous data points. It compares an observed data point with the median of past data points to obtain the distance between these data points. If the distance exceeds a specified threshold, the observed data point is marked as an outlier. In the signal correlation module, a gradual itemset mining algorithm is used to discover frequent covariations between signals. In the location correlation module, a heuristics algorithm based on offline correlation chains is used to extract a list of possible locations for each chain.

6) DYNAMIC META-LEARNING FOR FAILURE PREDICTION

Lan et al. [14] presented a dynamic meta-learning framework to predict failures in large scale systems. Their framework consists of two parts: (a) preprocessing system events in system logs and (b) generating failure patterns and triggering warnings. In the first part, the `Facility` field contained in RAS logs is used to divide system events into several high-level classifications. Then, the `Severity` and `Entity` fields contained in the RAS logs are used to divide the high-level classifications into 219 low-level event types. Temporal compression and spatial compression are used to remove duplicate system events. System events that contain the same `JobID` and `Location` values are coalesced into one entity. System events that contain the same `Entry Date` and `JobID` values and are close together within a predefined time window are removed. In the second part, three components are implemented: (a) a meta-learner, (b) a reviser and (c) a predictor. The meta-learner uses association rules, statistical and probability distribution methods to learn and reveal cause-and-effect relationships between system events. The reviser receives candidate rules generated by the meta-learner and uses the Receiver Operating Characteristics (ROC) technique to select optimal rules and discard suboptimal rules. The predictor monitors system events and triggers a failure warning when a rule is observed within a specified time window.

7) RESOURCE PREDICTION FOR DYNAMIC WORKLOADS IN CLOUDS

RPTCN is a method for predicting resource usage on cloud computing systems [15]. It consists of four steps: (a) data preprocessing, (b) correlation analysis, (c) data expansion and (d) method design. In the first step, feature scaling is applied to the data to reduce bias during model training. To scale the features, the MinMax scaling technique is used to rescale feature values (e.g., CPU usage, memory usage, disk I/O usage) so that the feature values range between 0 and 1. In the second step, correlations between a dependent feature and independent features are ranked. Then, the top half of the ranked features are selected as the input data. In the third step, the selected features are expanded to create additional features by extending each feature to three time sequences. In the fourth step, a Temporal Convolution Network (TCN), fully connection layer and attention mechanism are integrated. The TCN takes as its input, a time series sequence and processes the data through a network of hidden layers. The fully connected layer combines the features extracted by the previous hidden layer and activates the output neurons to produce a probability value. Then, an attention mechanism adjusts the weights applied at the hidden layer and output layer at different times.

8) LINKING ANOMALIES IN RESOURCE USE WITH SYSTEM FAILURES

Chuah et al. [22] presented a failure diagnostics approach called *ANCOR*. FDiag (see Section V-A4) and ANCOR use Pearson correlation to extract a list of correlated log-events. Differently to FDiag, ANCOR evaluates several feature extraction algorithms and identifies nodes that exhibited anomalous behaviour. ANCOR consists of four modules: (a) an anomaly extractor, (b) a message types extractor, (c) a time-bin correlator and (d) an event sequence constructor. The anomaly extractor module uses PCA, ICA and Mahalanobis Distance algorithms to identify nodes and jobs that exhibited anomalous workload patterns and outputs two lists containing the anomalous jobs and nodes. The message types extractor module receives these lists, obtains their system logs and extracts message types. The time-bin correlation module uses Pearson correlation to identify strongly positive correlated message types and a correlation threshold value of 0.75 is used. If the correlation strength between two message types is greater than or equals to 0.75, these message types are extracted and stored in a list of strongly positive correlated message types. The event sequence construction module extracts sequences of correlated log-events.

9) USING PARTIAL CORRELATION TO DIAGNOSE FAILURES IN CLUSTER SYSTEMS

IFADE is a novel diagnostics framework for identifying previously unknown causes of cluster system failures [23]. ANCOR and IFADE use feature extraction algorithms. Differently to ANCOR, FDiag and CORRMEXT, IFADE uses partial correlation. Partial correlation is a statistical technique for measuring the relationship strength between two variables while controlling for the effect that one or more variables have on the pair of variables. IFADE consists of three modules: (a) a data preprocessor, (b) a features extractor and (c) a partial correlator. The data preprocessor module extracts message types and resource use counters from system logs and resource use data respectively. Then, counts of message types and counts of resource use counters by time are converted into a message types data matrix and resource use data matrix respectively. The features extractor module uses PCA, ICA, Non-linear PCA and Kernel PCA algorithms to identify important messages types and resource use counters. Important message types or resource use counters are message types or resource use counters that were assigned high absolute scores by PCA, ICA, Non-linear PCA or Kernel PCA. The partial correlation module obtains correlations of pairs of message types or correlations of pairs of resource use counters after controlling for a third message type or resource use counter. It uses a correlation threshold value of 0.8. If the correlation strength between two message types or two resource use counters is greater than or equals to 0.8, these message types or resource use counters are extracted and stored in a list of strongly positive correlated message types or strongly positive correlated resource use counters.

VI. LOG-CORRELATION TOOLS QUALITY MODEL

In this section, we describe our quality model for evaluating the LogCTs. In our quality model, we define four quality attributes that we have identified through our systematic literature review. The quality attributes are spurious correlations, threshold settings, outliers in the data and missing data. Next, we present a typical question, a definition and a classification criterion for each quality attribute.

A. SPURIOUS CORRELATIONS

1) TYPICAL QUESTION

Can the LogCT identify relationships between two or more log-events that appears to be causal but is not?

2) DEFINITION

In statistics, a spurious relationship or spurious correlation is defined as a mathematical relationship in which two or more variables are associated but not causally related, due to either the presence of one or more confounding variables or purely by coincidence [28].

3) CLASSIFICATION CRITERIA

We analyzed the approach of each LogCT. If a LogCT did not identify spurious correlations of log-events, we classify it as ''Likely cause''. In contrast, if a LogCT considered spurious correlations of log-events, we classify it as ''Potential true cause''.

4) DOMAIN

Likely cause, Potential true cause.

B. CORRELATION THRESHOLD SETTINGS

1) TYPICAL QUESTION

How are the correlation threshold settings determined by the LogCT?

2) DEFINITION

A threshold value is used for removing redundant log-events, extracting strongly correlated log-events or identifying similarities between log-events. For example, FDiag [17] uses a predefined correlation coefficient value to extract a list of strongly correlated log-messages. Conversely, as an example, Zheng et al. [19] use a temporal-spatial filtering policy to identify and remove redundant log-events.

3) CLASSIFICATION CRITERIA

We analyzed how each LogCT selects the threshold value. If a LogCT uses a given, fixed or specified value as the threshold value, we classify it as ''Predefined''. If a LogCT uses a filtering policy to remove redundant log-events, we classify it as ''Adjustable''. If a LogCT did not describe how it obtains the threshold value, we classify it as ''Not specified''.

4) DOMAIN
Predefined, Adjustable, Not specified.

C. OUTLIERS IN THE DATA
1) TYPICAL QUESTION
Is the LogCT robust against outliers in the data?

2) DEFINITION
In statistics, an outlier is defined as a data point that differs significantly from other data points [51]. Kim et al. [29] showed that when outliers are present in both variables at the same time, these ''coincidental outliers'' can cause some uncorrelated or weakly correlated variables to show up as strongly correlated variables. If a LogCT did not remove outliers in the data, it may lead to an incorrect diagnosis or prediction of a failure.

3) CLASSIFICATION CRITERIA
We analyzed the approach in each LogCT. If a LogCT is evaluated on noisy data and showed to be robust in the presence of outliers, we classify it as ''Robust in the presence of outliers''. If a LogCT is evaluated on noisy data and showed to degrade gracefully in the presence of outliers, we classify it as ''Degrade in the presence of outliers''. If a LogCT is not evaluated on noisy data, we classify it as ''Not robust against outliers''.

4) DOMAIN
Robust against outliers, degrade in the presence of outliers, not robust against outliers.

D. MISSING DATA
1) TYPICAL QUESTION
Can the LogCT detect missing log-events?

2) DEFINITION
In statistics, missing data is defined as ''*a data value that is not stored for a variable in the observation of interest*'' [52]. A dataset containing cells that are empty presents two problems. The first problem is that statistical power is reduced, which refers to an incorrect rejection of the null hypothesis when it is false. The second problem is selection bias, which refers to a subset of data samples that are excluded from the study due to a flaw in the selection process of the data samples. For example, Rouillard [12] showed that if the system log rotation mechanism crashed but failed to report it, the system logs in the partition fills up over time and leads to an out-of-disk space error, causing the loss of system logs.

3) CLASSIFICATION CRITERIA
We analyzed the approach in each LogCT. If a LogCT is evaluated on a dataset which contains empty cells, we classify it as ''Robust against message loss''. If a LogCT provides a mechanism for handling missing log-events but was not evaluated on a dataset containing empty cells, we classify it as ''Potential for handling message loss''. If a LogCT is not evaluated on a dataset containing empty cells, we classify it as ''Not robust against message loss''.

4) DOMAIN
Robust against message loss, potential for handling message loss, not robust against message loss.

VII. LOG-CORRELATION TOOLS COMPARISON
In this section, we present a comparison of the log-correlation tools reviewed in Section V. We use the quality attributes described in Section VI to compare these log-correlation tools. A summary is given in Table 2. Next, we summarize the results reported in the LogCTs articles, give our observations and provide directions for future research.

A. SPURIOUS CORRELATIONS
From Table 2, we observed that SEC [12], LogMaster [13], RPTCN [15], SIG [3], FDiag [17], Baler [18], UiLog [20], CORRMEXT [21], ANCOR [22] and the approaches presented by Gainaru et al. [4], Lan et al. [14], Fu et al. [16], and Zheng et al. [19] performed direct correlation of log-events. In contrast, IFADE performed indirect correlation of log-events [23].

1) RESULTS
Taeret et al. [18] evaluated Baler with four log-clustering tools and showed that Baler identified several message patterns associated with out-of-memory conditions and memory errors. FDiag identified sequences of correlated messages associated with the Lustre file-system Evict/RPC protocol [17]. ANCOR detected nodes that exhibited anomalous behaviour and identified sequences of correlated messages associated with compute node soft lockups [22]. Oliner et al. [3] showed that SIGs identified bugs that caused two autonomous vehicles to swerve and identified a non-performance bug on a large supercomputer. Zheng et al. [19] identified the cause of hardware and system software failures on a BlueGene/L supercomputer. CORRMEXT identified patterns of error messages and obtained the percentage of success and failure of error recovery protocols [21]. Rouillard [12] showed how SEC can be used to analyze the normal and abnormal behaviour of the sendmail protocol. They provided a PERL script and applied it on the system logs obtained from a cluster system. Zou et al. [20] showed that (a) their fault log-analysis module achieved high precision rates for 12 types of faults and (b) the fault log-correlation analysis module decreased the collision error rates to about 20% and maintained a low truncation rate of less than 20%. IFADE evaluated partial correlation with direct correlation algorithms and showed that partial correlation identified two previously unknown causes of compute node soft lockups, which are (a) memory data updates and (b) corrupted memory pointers [23].

Gainaru et al. [4] evaluated their approach with two existing prediction methods and showed that their approach

TABLE 2. Log-Correlation Tools Quality Attributes.

Tool	Goal	Method	Dataset	Spurious correlation	Threshold settings	Handle outliers	Handle missing data
SEC [12]	Monitoring	Rule-based matching	Syslogs	No	Adjustable	No	Yes
LogMaster [13]	Failure prediction	Frequent itemset-mining, ECG	Hadoop logs, BlueGene/L, HPC cluster logs	No	Predefined	No	No
A. Gainaru *et al.* [4]	Failure prediction	Signal processing, Message clustering, Gradual itemset mining	BlueGene/L	No	Predefined	Yes	No
Z. Lan *et al.* [14]	Failure prediction	Association rules, Statistical rules, Probability distribution	BlueGene/L	No	Adjustable	No	No
RPTCN [15]	Resource prediction	Pearson-correlation, TCN	Cluster traces	No	Predefined	No	No
X. Fu *et al.* [16]	Failure diagnosis & prediction	Graph mining	Hadoop logs, BlueGene/L, HPC cluster logs	No	Predefined	No	No
SIG [3]	Failure diagnosis	KL divergence, Pearson correlation	Autonomous vehicles, HPC cluster logs	No	Adjustable	Yes	Yes
FDiag [17]	Failure diagnosis	Pearson correlation, Episode construction	Syslogs	No	Predefined	No	No
Baler [18]	Failure diagnosis	Levenshtein distance	HPC cluster logs	No	Predefined	No	No
Z. Zheng *et al.* [19]	Failure diagnosis	Temporal-spatial filtering, DBSCAN	BlueGene/L	No	Adjustable	No	No
UiLog [20]	Failure diagnosis	Invariants mining, TFIDF, DBSCAN, Levenshtein distance	Cloud platform logs	No	Adjustable	No	No
CORRMEXT [21]	Error propagation	Pearson correlation, Spearman-Rank correlation	Rationalized logs, Syslogs, Resource use data	No	Predefined	No	No
ANCOR [22]	Failure diagnosis	Feature extraction, Pearson correlation, Episode construction	Rationalized logs, Resource use data	No	Predefined	Yes	No
IFADE [23]	Failure diagnosis	Feature extraction, Partial correlation	Rationalized logs, Syslogs, Resource use data	Yes	Predefined	No	No

outperformed those prediction methods in terms of the precision and recall. Lan et al. [14] showed that their dynamic meta-learning approach outperformed a static meta-learner in terms of the precision and recall. The authors of RPTCN [15] compared it with four commonly used time series prediction algorithms and showed that RPTCN outperformed those algorithms in terms of the lowest mean squared error and mean absolute error. Fu et al. [13] showed that LogMaster achieved a high average precision on the system logs of three cluster systems and confirmed that correlations between failure and non-failure events are crucial for predicting system failures accurately. Subsequently, Fu et al. [16] enhanced LogMaster and applied it on three real world system logs and showed that (a) the failure prediction step achieved high precision and recall rates on all three system logs and outperformed three existing prediction algorithms and (b) the failure diagnostics step achieved comparable precision and recall rates with respect to those prediction algorithms.

2) OBSERVATION

The reviewed log-correlation tools are effective in identifying the likely cause of a system failure or predicting system failures with high accuracy. However, if the correlated log-events are not causally linked due to the presence of one or more confounding log-events, then this might lead to a wrong diagnosis or a wrong prediction [27].

3) PROMISING RESEARCH DIRECTION

There are few works that identify spurious correlations which may lead to a false diagnosis or false prediction. In statistics, spurious correlations are used to rule out correlations that do not represent causal relationships [28]. Spurious correlations can be identified by controlling for the effect of one or more confounding variables. Partial correlation is a statistical technique for measuring the relationship between two variables while controlling for the effect of one or more other variables [53]. Thus, researchers could investigate further correlation techniques to reduce

the probability of making a wrong diagnosis or a wrong prediction. While current LogCTs are effective in diagnosing system failures or predicting system failures, tools which can improve failure diagnosis or failure prediction could be a valuable addition to contemporary failure management systems for large supercomputers (e.g., [24]).

B. CORRELATION THRESHOLD SETTINGS

From Table 2, we observed that SEC [12], SIGs [3], UiLog [20] and the approaches presented by Lan et al. [14] and Zheng et al. [19] provided adjustable threshold values. In contrast, LogMaster [13], RPTCN [15], FDiag [17], Baler [18], CORRMEXT [21], ANCOR [22], IFADE [23] and the approaches presented by Gainaru et al. [4] and Fu et al. [16] used a predefined threshold value.

1) RESULTS

Oliner et al. [3] used two parameters for computing a SIG. These parameters can be adjusted to explore their impact on a SIG. Rouillard [12] used the counts of unique events to determine when a rule containing a threshold correlation value is satisfied. Zou et al. [20] used different time-windows to identify correlations of log-events and assign these log-events to a specific fault type. Zheng et al. [19] used different similarity scores to distinguish application failures from hardware failures.

Gainaru et al. [4] used predefined threshold values for each signal obtained from the preprocessing step of their approach. Each threshold value represents the normal behaviour of one event type. Taerat et al. [18] used two parameters to determine the similarity between log-events. They conducted experiments with different values on several system logs and showed that a threshold value of 0.3 produced very few over-grouped clusters. Fu et al. [13] defined a support count and posterior count for computing a confidence measure and used a predefined value to determine the support count for representing a cluster of frequently occurring log-events. Chen et al. [15] used the top-half of all ranked performance metrics as input data to the temporal convolution network. To filter duplicate events, Lan et al. [14] implemented an iterative approach where each value is tested and increased until the log-compression rate is stable. They showed that a threshold value of 300 seconds produced a 98% compression rate for the IBM BlueGene/L logs. FDiag [17], ANCOR [22], CORRMEXT [21], and IFADE [23] used a fixed correlation coefficient value of 0.75 to identify the strongly positive correlated log-messages, but the strongly positive correlation coefficient value range from 0.75 to 1.

2) OBSERVATION

Determining the impact that the correlation threshold settings will have on the LogCTs in diagnosing system failures or predicting system failures is challenging, especially when different techniques are implemented and these techniques require different ranges of values.

3) PROMISING RESEARCH DIRECTION

There are few works that assess the impact of tuning the correlation threshold settings for diagnosing system failures [44] or predicting system failures. Serafini et al. [44] showed that the completeness and correctness of failure diagnosis is influenced by tuning the parameters in various fault models during design time. Thus, researchers could study how the parameters in correlation models can be tuned to improve failure diagnosis or failure prediction.

C. OUTLIERS IN THE DATA

From Table 2, we observed that SIGs [3], ANCOR [22] and the approach presented by Gainaru et al. [4] handled outliers in the system logs. In contrast, SEC [12], UiLog [20], LogMaster [13], RPTCN [15], FDiag [17], Baler [18], CORRMEXT [21], IFADE [23] and the approaches presented by Lan et al. [14], Zheng et al. [19], and Fu et al. [16] assumed that there are no outliers in the system logs.

1) RESULTS

Gainaru et al. [4] implemented outlier detection in their analysis modules. They computed the median of past data points and compared it with the current data point. If the distance between the median value and the current data point exceeds a specified threshold, the current data point is labelled as an outlier. Then, a replacement value for the data point is proposed. Oliner et al. [3] quantified the behaviour of system components in terms of the anomaly signal. They defined an anomaly signal as a signal that takes a value near to the mean of its distribution, so when the distance between a data point and the mean of the anomaly signal increases, an anomaly in the system component's behaviour is detected. They used these anomaly signals to construct a Structure-of-Influence Graph (SIG) and showed that SIGs are robust against measurement noise. ANCOR applied feature extraction techniques to identify nodes that exhibited anomalous behaviour and diagnose the cause of compute node soft lockups on those anomalous nodes [22].

2) OBSERVATION

Outliers in the system logs have been used to distinguish normal system behaviour from abnormal system behaviour [4] and to diagnose the cause of system failures [3], [22]. However, few works have investigated the impact of outliers on the accuracy of failure diagnosis or failure prediction.

3) PROMISING RESEARCH DIRECTION

Researchers could investigate novel correlation approaches which are robust against outliers, for example using a rank correlation algorithm rather than Pearson correlation [31].

D. MISSING DATA

From Table 2, we observed that SEC [12] and SIGs [3] handled missing log-events. In contrast, LogMaster [13], RPTCN [15], FDiag [17], Baler [18], CORRMEXT [21],

ANCOR [22], IFADE [23], UiLog [20] and the approaches presented by Lan et al. [14], Zheng et al. [19], Gainaru et al. [4], and Fu et al. [16] assumed that all the log-events are contained in the system logs.

1) RESULTS

Rouillard [12] showed an example on how to handle the problem of missing log-events. They wrote a PERL script that generated cron job log-events regularly and matched these generated log-events to cron job log-events in the system logs. If there are no cron job log-events in the system logs, then it indicates that the cron job process had crashed. Moreover, they pointed out that it is difficult to determine when the external log-event generating process had failed. Oliner et al. [3] treated message loss as another form of noise. They presented an experiment that produced output messages at t_1, t_2 and t_3 and showed that SIGs are robust against uniform message loss.

2) OBSERVATION

There are few articles that investigated the impact of missing log-events on the accuracy of failure diagnosis or failure prediction.

3) PROMISING RESEARCH DIRECTION

Researchers could implement novel approaches to test for the presence of specific log-events, for example generate Lustre file-system Evict/RPC protocol events and match those generated log-events to the log-events in the system logs to determine if the Lustre Evict/RPC protocol had crashed. Researchers could also investigate novel statistical methods for handling missing data, for example listwise deletion, pairwise deletion, mean substitution, regression imputation or sensitivity analysis [52].

VIII. THREATS TO VALIDITY

We have identified the following threats to validity: (a) construct validity threat, (b) internal validity threat and (c) external validity threat.

A construct validity threat is concerned with the observations with respect to the established theory. We proposed a quality model that covered the important attributes of LogCTs. We identified the attributes of spurious correlation, threshold settings, outliers in the data and missing data from a detailed review of 46 representative articles that were obtained through a systematic literature review process. As the 46 articles are representative of LogCTs, we believe that there is no threat to the accuracy of our results, apart from the threat to the design of a systematic literature review. We may have missed some relevant articles, so to address this issue we followed established guidelines on how to conduct a systematic literature review [7].

An internal validity threat is concerned with the factors that might influence the results presented in this paper. These factors include the selection of articles and the characteristics of LogCTs. On the selection of articles, we may have missed relevant articles on LogCTs. To address this issue, we applied a systematic literature review process [7] and searched multiple scholarly databases which are the main sources of research articles. These databases include the digital libraries of the IEEE, ACM, Springer and Elsevier. Furthermore, we conducted both forward and backwards snowballing to mitigate the risk of missed articles. On the characteristics of LogCTs, our analysis of the characteristics of LogCTs are based on the approach and results provided in the selected articles. We did not review LogCTs for security because they are beyond the scope of this paper, nor check the source code or implement the reviewed LogCTs as it would require a significant amount of resources out of the scope of this paper.

An external validity threat is concerned with the generalizability of the results. Our conclusions are based on 14 LogCTs and may not generalize to all LogCTs. As the LogCTs we reviewed are widely deployed in large cluster systems, we believe that our results are representative for most LogCTs. Regardless of the generalizability of our results, we showed that our quality model can be used to compare any LogCT.

IX. CONCLUSION AND FUTURE WORK

We presented a systematic review of literature on LogCTs. From 3,758 initial articles, we reviewed 46 representative articles in detail and identified the important attributes associated with the LogCTs. We proposed a quality model containing four quality attributes comprised of spurious correlation, threshold settings, outliers in the data and missing data, and used our quality model to compare and evaluate 14 LogCTs. We showed that there is not one LogCT that fulfilled all the four quality attributes.

System designers can use the results in Section VII to decide which LogCT is relevant to their specific use case. Researchers can use our quality model to compare and evaluate state-of-the-art LogCTs and also use our recommendations to enhance existing LogCTs or develop new LogCTs. We have made several recommendations for future research. They are (a) design novel correlation approaches which identify spurious correlations to reduce the risk of making a wrong diagnosis or false prediction, (b) study how the parameters of correlation models can be tuned to improve diagnostics or prediction accuracy, (c) design novel correlation approaches which are robust against outliers in the system logs and (d) implement novel approaches for handling missing log-events in the system logs.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their constructive feedback, which helped improve the paper significantly.

REFERENCES

[1] *IEEE Standard for Information Technology—Portable Operating System Interface (POSIX(R)*, IEEE Standards 1003.1-2001, 2001.

[2] A. Miranskyy, A. Hamou-Lhadj, E. Cialini, and A. Larsson, "Operational-log analysis for big data systems: Challenges and solutions," *IEEE Softw.*, vol. 33, no. 2, pp. 52–59, Mar. 2016.
[3] A. J. Oliner, A. V. Kulkarni, and A. Aiken, "Using correlated surprise to infer shared influence," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2010, pp. 191–200.
[4] A. Gainaru, F. Cappello, M. Snir, and W. Kramer, "Fault prediction under the microscope: A closer look into HPC systems," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2012, p. 77.
[5] Q. Guan, D. Smith, and S. Fu, "Anomaly detection in large-scale coalition clusters for dependability assurance," in *Proc. Int. Conf. High Perform. Comput.*, Dec. 2010, pp. 1–10.
[6] J. C. Browne et al., "Comprehensive, open-source resource usage measurement and analysis for HPC systems," *Concurrency Comput., Pract. Exper.*, vol. 26, no. 13, pp. 2191–2209, Sep. 2014.
[7] K. S. Khan, R. Kunz, J. Kleijnen, and G. Antes, "Five steps to conducting a systematic review," *JRSM*, vol. 96, no. 3, pp. 118–121, Mar. 2003.
[8] S. He, P. He, Z. Chen, T. Yang, Y. Su, and M. R. Lyu, "A survey on automated log analysis for reliability engineering," *ACM Comput. Surveys*, vol. 54, no. 6, pp. 1–37, Jul. 2021.
[9] S. P. Kavulya, K. Joshi, F. D. Giandomenico, and P. Narasimhan, "Failure diagnosis of complex systems," in *Resilience Assessment and Evaluation of Computing Systems*, K. Wolter, A. Avritzer, M. Vieira, and A. van Moorsel, Eds. Berlin, Germany: Springer.
[10] F. Salfner, M. Lenk, and M. Malek, "A survey of online failure prediction methods," *ACM Comput. Surveys*, vol. 42, no. 3, pp. 1–42, Mar. 2010.
[11] D. El-Masri, F. Petrillo, Y.-G. Guéhéneuc, A. Hamou-Lhadj, and A. Bouziane, "A systematic literature review on automated log abstraction techniques," *Inf. Softw. Technol.*, vol. 122, Jun. 2020, Art. no. 106276.
[12] J. P. Rouillard, "Real-time log file analysis using the simple event correlator (SEC)," in *Proc. 18th USENIX Conf. Syst. Admin.*, 2004, pp. 133–150.
[13] X. Fu, R. Ren, J. Zhan, W. Zhou, Z. Jia, and G. Lu, "LogMaster: Mining event correlations in logs of large-scale cluster systems," in *Proc. IEEE 31st Symp. Reliable Distrib. Syst.*, Oct. 2012, pp. 71–80.
[14] Z. Lan, J. Gu, Z. Zheng, R. Thakur, and S. Coghlan, "A study of dynamic meta-learning for failure prediction in large-scale systems," *J. Parallel Distrib. Comput.*, vol. 70, no. 6, pp. 630–643, Jun. 2010.
[15] W. Chen, C. Lu, K. Ye, Y. Wang, and C.-Z. Xu, "RPTCN: Resource prediction for high-dynamic workloads in clouds based on deep learning," in *Proc. IEEE Int. Conf. Cluster Comput. (CLUSTER)*, Sep. 2021, pp. 59–69.
[16] X. Fu, R. Ren, S. A. McKee, J. Zhan, and N. Sun, "Digging deeper into cluster system logs for failure prediction and root cause diagnosis," in *Proc. IEEE Int. Conf. Cluster Comput. (CLUSTER)*, Sep. 2014, pp. 103–112.
[17] E. Chuah, S.-H. Kuo, P. Hiew, W.-C. Tjhi, G. Lee, J. Hammond, M. T. Michalewicz, T. Hung, and J. C. Browne, "Diagnosing the root-causes of failures from cluster log files," in *Proc. Int. Conf. High Perform. Comput.*, Dec. 2010, pp. 1–10.
[18] N. Taerat, J. Brandt, A. Gentile, M. Wong, and C. Leangsuksun, "Baler: Deterministic, lossless log message clustering tool," *Comput. Sci. Res. Develop.*, vol. 26, nos. 3–4, pp. 285–295, Jun. 2011.
[19] Z. Zheng, L. Yu, Z. Lan, and T. Jones, "3-dimensional root cause diagnosis via co-analysis," in *Proc. 9th Int. Conf. Autonomic Comput. (ICAC)*, 2012, pp. 181–190.
[20] D.-Q. Zou, H. Qin, and H. Jin, "UiLog: Improving log-based fault diagnosis by log analysis," *J. Comput. Sci. Technol.*, vol. 31, no. 5, pp. 1038–1052, Sep. 2016.
[21] E. Chuah, A. Jhumka, S. Alt, D. Balouek-Thomert, J. C. Browne, and M. Parashar, "Towards comprehensive dependability-driven resource use and message log-analysis for HPC systems diagnosis," *J. Parallel Distrib. Comput.*, vol. 132, pp. 95–112, Oct. 2019.
[22] E. Chuah, A. Jhumka, S. Narasimhamurthy, J. Hammond, J. C. Browne, and B. Barth, "Linking resource usage anomalies with system failures from cluster log data," in *Proc. IEEE 32nd Int. Symp. Reliable Distrib. Syst.*, Sep. 2013, pp. 111–120.
[23] E. ChuahM, A. Jhumka, S. Alt, R. T. Evans, and N. Suri, "Failure diagnosis for cluster systems using partial correlations," in *Proc. IEEE Int. Conf Parallel Distrib. Process. Appl., Big Data Cloud Comput., Sustain. Comput. Commun., Social Comput. Netw. (ISPA/BDCloud/SocialCom/SustainCom)*, Sep. 2021, pp. 1091–1101.
[24] J. L. Hammond, T. Minyard, and J. Browne, "End-to-end framework for fault management for open source clusters: Ranger," in *Proc. TeraGrid Conf.*, 2010, pp. 1–6.
[25] A. Agresti and C. Franklin, *Statistics: The Art and Science of Learning From Data*. Hoboken, NJ, USA: Prentice-Hall, 2009.
[26] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Reading, MA, USA: Addison-Wesley, 2006.
[27] A. Goudarzi, D. Arnold, D. Stefanovic, K. B. Ferreira, and G. Feldman, "A principled approach to HPC event monitoring," in *Proc. 5th Workshop Fault Tolerance for HPC at eXtreme Scale*. New York, NY, USA: Association for Computing Machinery, Jun. 2015, pp. 3–10.
[28] H. A. Simon, "Spurious correlation: A causal interpretation," *J. Amer. Stat. Assoc.*, vol. 49, no. 267, pp. 467–479, 1954.
[29] Y. Kim, T.-H. Kim, and T. Ergün, "The instability of the Pearson correlation coefficient in the presence of coincidental outliers," *Finance Res. Lett.*, vol. 13, pp. 243–257, May 2015.
[30] V. R. Patel and R. G. Mehta, "Impact of outlier removal and normalization approach in modified K-means clustering algorithm," *Int. J. Comput. Sci. Issues*, vol. 8, no. 5, p. 331, 2011.
[31] F. Salfner, P. Troger, and S. Tschirpke, "Cross-core event monitoring for processor failure prediction," in *Proc. Int. Conf. High Perform. Comput. Simul.*, Jun. 2009, pp. 67–73.
[32] P. L. Roth, "Missing data: A conceptual review for applied psychologists," *Pers. Psychol.*, vol. 47, no. 3, pp. 537–560, 2010.
[33] M. Moh, S. Pininti, S. Doddapaneni, and T.-S. Moh, "Detecting web attacks using multi-stage log analysis," in *Proc. IEEE 6th Int. Conf. Adv. Comput. (IACC)*, Feb. 2016, pp. 733–738.
[34] A. Fourney, R. Mann, and M. Terry, "Characterizing the usability of interactive applications through query log analysis," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, May 2011, pp. 1817–1826.
[35] D. Galar and U. Kumar, "Chapter 5—Diagnosis," in *eMaintenance*, D. Galar and U. Kumar, Eds. New York, NY, USA: Academic, 2017, pp. 235–310.
[36] N. Taerat, N. Nakisinehaboon, C. Chandler, J. Elliot, C. Leangsuksun, G. Ostrouchov, and S. L. Scott, "Using log information to perform statistical analysis on failures encountered by large-scale HPC deployments," in *Proc. High Availability Perform. Comput. Workshop*, 2008, pp. 1–6.
[37] A. Abu-Samah, M. Shahzad, E. Zamai, and A. Said, "Failure prediction methodology for improved proactive maintenance using Bayesian approach," *IFAC-PapersOnLine*, vol. 48, no. 21, pp. 844–851, 2015.
[38] M. A. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, Doctor Philosophy, Univ. Waikato, Hamilton, New Zealand, 1999.
[39] S. Di, R. Gupta, M. Snir, E. Pershey, and F. Cappello, "LOGAIDER: A tool for mining potential correlations of HPC log events," in *Proc. 17th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*, May 2017, pp. 442–451.
[40] K. A. Alharthi, A. Jhumka, S. Di, F. Cappello, and E. Chuah, "Sentiment analysis based error detection for large-scale systems," in *Proc. 51st Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2021, pp. 237–249.
[41] B. Schroeder and G. Gibson, "The computer failure data repository (CFDR): Collecting, sharing and analyzing failure data," in *Proc. ACM/IEEE Conf. Supercomput. (SC)*, Nov. 2006, p. 154-es.
[42] S. Bagchi, T. Evans, R. Kumar, R. Kalyanam, S. Harrell, C. Ellis, and C. Song. (2015). *FRESCO: Job Failure and Performance Data Repository From Purdue University*. Engineering Purdue University. [Online]. Available: https://www.datadepot.rcac.purdue.edu/sbagchi/fresco
[43] A. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," in *Proc. 37th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2007, pp. 575–584.
[44] M. Serafini, A. Bondavalli, and N. Suri, "On-line diagnosis and recovery: On the choice and impact of tuning parameters," *IEEE Trans. Dependable Secure Comput.*, vol. 4, no. 4, pp. 295–312, Oct. 2007.
[45] S. E. Hansen and E. T. Atkins, "Automated system monitoring and notification with SWATCH," in *Proc. USENIX Large Installation Syst. Admin. Conf. (LISA)*, 1993, pp. 101–108.
[46] R. Vaarandi, "SEC—A lightweight event correlation tool," in *Proc. IEEE Workshop IP Operations Manage.*, Dec. 2002, pp. 111–115.
[47] A. J. Oliner and A. Aiken, "Online detection of multi-component interactions in production systems," in *Proc. IEEE/IFIP 41st Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2011, pp. 49–60.
[48] E. Chuah, A. Jhumka, J. C. Browne, B. Barth, and S. Narasimhamurthy, "Insights into the diagnosis of system failures from cluster message logs," in *Proc. 11th Eur. Dependable Comput. Conf. (EDCC)*, Sep. 2015, pp. 1–8.

[49] R. T. Evans, J. C. Browne, and W. L. Barth, "Understanding application and system performance through system-wide monitoring," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, May 2016, pp. 1702–1710.
[50] E. Chuah, A. Jhumka, S. Alt, J. J. Villalobos, J. Fryman, W. Barth, and M. Parashar, "Using resource use data and system logs for HPC system error propagation and recovery diagnosis," in *Proc. IEEE Int. Conf Parallel Distrib. Process. Appl., Big Data Cloud Comput., Sustain. Comput. Commun., Social Comput. Netw. (ISPA/BDCloud/SocialCom/SustainCom)*, Dec. 2019, pp. 1–10.
[51] D. M. Hawkins, "Identification of outliers," in *Monographs on Statistics and Applied Probability*, vol. 11. Dordrecht, The Netherlands: Springer, 1980.
[52] J. W. Graham, *Missing Data: Analysis and Design*. Berlin, Germany: Springer, 2012.
[53] F. V. Waugh, "The computation of partial correlation coefficients," *J. Amer. Stat. Assoc.*, vol. 41, no. 236, pp. 543–546, Dec. 1946.

EDWARD CHUAH received the Ph.D. degree in computer science from the University of Warwick, U.K., in July 2020. Before his Ph.D., he worked in Singapore as a Research Engineer, a Software Engineer, and a Lecturer (1.5 years in Software Engineering, eight years in Research and Development, and two years in Teaching). After his Ph.D., he commenced his research in network security as a Postdoctoral Researcher at Lancaster University, U.K. He is currently a Lecturer with the Computing Science Department, University of Aberdeen, U.K. He is also a Former Doctoral Student with The Alan Turing Institute, U.K. His research interests are in large-scale systems dependability, HPC reliability (failure diagnosis, failure prediction, error propagation and error detection), network security (attacks identification), and data analysis. His work involves processing large volumes of real data to generate new insights into a distributed system to enhance its reliability and security.

ARSHAD JHUMKA received the Ph.D. degree in computer science from TU-Darmstadt, Germany, in 2003. He is currently a Reader with the Department of Computer Science, University of Warwick, U.K. He was previously a fellow of The Alan Turing Institute, U.K. In 2005, he joined the University of Warwick as a Lecturer. His current research interests include the intersection of large-scale distributed systems (e.g., wireless sensor networks, cluster systems), embedded and/or cyber physical systems, and dependability. He has published over 70 technical papers in leading conferences and journals. His papers have garnered two best paper awards from IEEE DSN, in 2001, and IEEE HASE, in 2002, and three of his papers were also best paper award candidates. He is a member of several TPCs including Mobiwac, SSS, PCAC, and DEPEND.

MIROSLAW MALEK (Life Senior Member, IEEE) received the M.Sc. degree in electrical engineering (electronics) from the Technical University of Wroclaw, Poland, and the Ph.D. degree in computer science from the Technical University of Wroclaw, in 1975.

In 1977, he was a Visiting Scholar at the Department of Systems Design, University of Waterloo, Waterloo, ON, Canada, then an Assistant, an Associate, and a Full Professor at The University of Texas at Austin (1977–1994), where he was also a holder of the Bettie Margaret Smith and the Southwestern Bell Professorships in engineering. From 1994 to 2012, he was a Professor and the Holder of Chair in computer architecture and communication at the Department of Computer Science, Humboldt University, Berlin. He was a Visiting Professor at Stanford University, Università di Roma La Sapienza, Politecnico di Milano, Technical University in Vienna, New York University, Chinese University of Hong Kong, held the IBM Chair at Keio University, and also a Visiting Scientist at the Bell Laboratories and the IBM T. J. Watson Research Center. He is currently a Professor with the Faculty of Informatics, Advanced Learning and Research Institute (ALaRI), University of Lugano (USI). His research interests include dependable and secure architectures and services in parallel, cloud, distributed, and embedded computing environments with emphasis on failure prediction using machine learning and data analytics. He has participated in two pioneering parallel computer projects, contributed to the theory and practice of parallel network design, developed the comparison-based method for system diagnosis, co-developed comprehensive WSI and networks testing techniques, proposed the consensus-based framework for responsive (fault-tolerant, real-time) computer systems design and has made numerous other contributions, reflected in over 300 publications, and nine books.

Dr. Malek has organized, chaired, and a program committee member of numerous IEEE and ACM international conferences and workshops. He serves or served on the Editorial Board Member of the *ACM Computing Surveys*, *Journal of Parallel and Distributed Computing*, *Journal of Interconnection Networks*, and *Real-Time Systems* journal. He has supervised 30 Ph.D. dissertations (ten of his students are professors), three habilitations, and founded, organized, and co-organized numerous workshops and conferences. He is a Consultant to startup companies, government, and multinational corporations on technical and strategic directions and activities in information technology. He delivered numerous Keynote Addresses and taught short courses in Europe, Asia, and North America.

NEERAJ SURI (Senior Member, IEEE) holds the Distinguished Professorship and the Chair in cybersecurity at Lancaster University, U.K., where he co-directs the university-wide Security Institute. He previously held the Chair Professorship in dependable systems and software at TU Darmstadt, Germany. Following his Ph.D. at UMass-Amherst, he has held positions at AlliedSignal/Honeywell Research, Boston University, Saab Endowed Chair Professorship, multiple sabbaticals at Microsoft Research, and a visiting positions at The University of Texas at Austin, Academia Sincia, PolyU Hong Kong, and Technion. His research interests include the design, analysis, assessment of trustworthy (dependable and secure), cloud systems, and software. He is currently an Advocate of the data-centric system approach to cybersecurity and on the quantification of security. His research has received extensive trans-national funding from the U.S.: NSF/DARPA/DHS/ONR/AFOSR, the EC FP6/FP7/H2020 (CONCORDIA, ESCUDO-CLOUD, NECS, CIPSEC, SLA-READY, SPECS, ABC4TRUST, BIC, INDEXYS, COMIFIN, INSPIRE, INCO-TRUST, THINK-TRUST, GENESYS, DECOS, RESIST, DBENCH, NEXTTTA), German DFG/BMBF/DAAD/Loewe, Swedish SSF/VINNOA/NUTEK, Microsoft, IBM, Amazon, Apple, Google, Boeing, GM, NASA, Airbus, SAP, NEC, Hitachi, Saab, Volvo, and Daimler.

He is a member of IFIP WG 10.4 on Fault Tolerance and Dependability, and served on multiple US/EU/Asian industry and academic advisory boards for IBM, Intel, NASA, Uppsala University, and EC's RISEPTIS Board for Trust and Security, and a member of Microsoft's Trustworthy Computing Academic Advisory Board (TCAAB). His professional services span the Associate EIC for IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, an Editorial Board Member for IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, IEEE TRANSACTIONS ON BIG DATA, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, *ACM Computing Surveys*, and IEEE SECURITY & PRIVACY. He was a recipient of the NSF CAREER, Microsoft, and IBM Faculty Awards. He has PC-chaired multiple dependability conferences (e.g., DSN, ICDCS, SRDS, HASE). He is the Co-Organizer for the 2020 Dagstuhl seminar on Characterizing and Modeling Residual Software Bugs. He chaired the IEEE Technical Committee on Dependability and Fault Tolerance, and it's Steering Committee. He currently chairs the Steering Committee for the IEEE International Conference on Cloud Engineering.