# An End-to-end Log Management Framework for Distributed Systems

Pinjia He
Computer Science and Engineering Department, The Chinese University of Hong Kong, Hong Kong
Shenzhen Research Institute, The Chinese University of Hong Kong, China
pjhe@cse.cuhk.edu.hk

***Abstract*—Logs have been widely employed to ensure the reliability of distributed systems, because logs are often the only data available that records system runtime information. Compared with logs generated by traditional standalone systems, distributed system logs are often large-scale and of great complexity, invalidating many existing log management methods. To address this problem, the paper describes and envisions an end-to-end log management framework for distributed systems. Specifically, this framework includes strategic logging placement, log collection, log parsing, interleaved logs mining, anomaly detection, and problem identification.**

## I. Introduction

Distributed systems have become the core building block of modern IT industry, powering various daily use software, such as company websites, online shopping platforms, and video sharing services. Different from traditional standalone systems, distributed systems often run on a 24×7 basis providing services for millions of users globally. A transient downtime of distributed systems can lead to unavailability of the supported software or even enormous revenue loss [1], [2]. Thus, a reliable distributed system is highly in demand.

Logs, which record system runtime information, have been widely employed to ensure the reliability of distributed systems. Typically, logs are printed by the logging statements written by developers, which consist of constant text describing the system operations and parameters recording system runtime status. The rich information hidden in logs allows various reliability assurance tasks, including anomaly detection [3], program verification [4], problem identification [5], etc.

Compared with logs generated by traditional standalone systems, distributed system logs are often large-scale and of great complexity. In particular, a distributed system could produce 120∼200 log messages in one hour [6], which renders intelligent logging placement, log collection and log parsing necessary and non-trivial. To print necessary log messages to record system runtime information without incurring unintended performance overhead, we need strategic logging placement in the first place. Besides, due to the large scale, we can collect only a subset of logs in system runtime, and infer the whole context for further reliability assurance tasks. After collection, we require a log parser that can parse tons of raw log messages into structured events. Modern distributed systems are of great complexity, often containing hundreds of distributed components and supporting a large number of concurrent users [5]. The complexity brings challenges to the log-based anomaly detection and problem identification tasks. Specifically, the traditional methods based on manual inspection become labor-intensive and error-prone. Moreover, logs generated by distributed systems are sometimes interleaved (e.g., logs of concurrent threads without thread IDs), which makes reliability assurance for distributed systems even harder.

To address these problems, we envision an end-to-end log management framework, which aims at providing solutions for every single step of log-based reliability assurance for distributed systems, including strategic logging placement, log collection, log parsing, interleaved logs mining, anomaly detection, and problem identification.

## II. Framework and Challenges

As illustrated by Fig. 1, the proposed framework contains 6 parts. First, we need to provide logging placement suggestions to developers, which is non-trivial because currently there is a lack of rigorous specifications on developers' logging behaviors. Second, in system runtime, logs will be printed by logging statements, which demands log collection strategies to record the necessary subset of logs to avoid log volume explosion. This is challenging, because omitting crucial information may lead to inaccuracy of reliability assurance tasks. After log collection, we employ log parser to transform raw logs, which is often unstructured, into structured events. The log parser need to be both accurate and efficient to ensure normal execution of subsequent anomaly detection and problem identification. Due to the distributed environment, sometimes a technique that can automatically separate the interleaved logs is required, because manually understanding these logs are difficult and error-prone. Then, based on the non-interleaved structured events, we can conduct anomaly detection to find potential failures automatically. Moreover, the problem type can be further identified by problem identification techniques. Anomaly detection and problem identification are prohibitive using traditional manual approaches, because developers need to spot the problem from tons of logs generated by distributed systems. Besides, it is complicated for developers to manually analyze the logs of hundreds of distributed components.

## III. Approach

In this section, we explain the techniques proposed for the end-to-end log management framework in detail. Specifically,

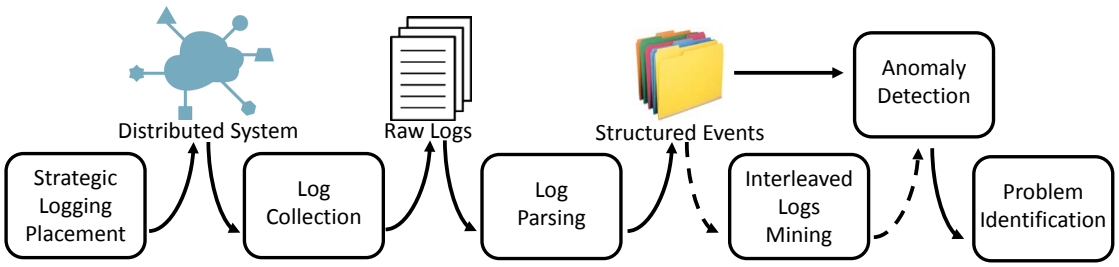


Fig. 1: The end-to-end log management framework

in each subsection, we introduce our proposed approach and/or planned work in the future.

**Strategic logging placement**: we design a "learning to log" approach to mine logging behaviors from codes written by expert developers, and further leverage them for actionable suggestions in programming [7]. Specifically, this approach suggests developers whether log or not in a specific coding block (e.g., try-catch block). To further improve the strategic logging placement, we plan to study the suggestion of exact position (i.e., the position in abstract syntax tree) for a logging statement. Specifically, we will consider program variables recorded by the logging statement and corresponding performance overhead.

**Log collection**: we propose a location-based approach to predict the QoS values (e.g., response time) of all Web services [8] based on the record of a subset of them. Specifically, our approach utilizes matrix factorization model with encoded location information on the recorded sparse QoS values. This work focus on the logs that only records the QoS values of the services in distributed systems. In the future, we intend to generalize this log collection method to common logs containing both constant text and parameters.

**Log parsing**: we have an experience report on four state-of-the-art offline log parsing methods in terms of their accuracy, efficiency, and effectiveness on subsequent reliability assurance tasks [9]. The source codes of the studied parsers have been released [10]. Besides, we design an online log parser, namely Drain, that can parse logs in a streaming and timely manner [11]. In the future, we will extend Drain by adding a dynamic and automated parameter tuning mechanism. Besides, we plan to collect more log data sets for log parsing and make them open-source.

**Interleaved logs mining**: we will conduct an evaluation study on the state-of-the-art interleaved logs mining approaches [12], [13]. Specifically, we will implement them and release them as an open-source toolkit. Besides, the accuracy and efficiency will be evaluated on both synthetic and real-world data sets. Moreover, we will summarize the advantages and disadvantages of the studied approaches.

**Anomaly detection**: we produce an experience report on six representative anomaly detection methods, including three unsupervised methods and three supervised methods [14]. The implementation of these methods have been released as a toolkit allowing ease of reuse [15]. We evaluate these methods on two publicly-available production log data sets, with a total of 15,923,592 log messages and 365,298 anomaly instances.

**Problem identification**: we plan to design a problem identification approach. In this approach, we cluster the structure events of historic problems into different groups based on the event occurrence frequency. Then, in system runtime, we calculate the similarity between the incoming events and different problem groups for identification. Different from the state-of-the-art approach [5], we mine invariant rules among different log events and employ PCA to find crucial features.

## ACKNOWLEDGMENT

The author is supervised by Prof. Michael R. Lyu.

## REFERENCES

[1] Facebook loses $24,420 a minute during outages (http://algerian–news.blogspot.hk/2014/10/facebook-loses-24420-minute-during.html).
[2] Disruption in amazon's cloud service ripples through internet (http://www.reuters.com/article/us-amazon-com-aws-outages-iduskbn1672e2).
[3] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordon, "Detecting large-scale system problems by mining console logs," in *SOSP'09: Proc. of the ACM Symposium on Operating Systems Principles*, 2009.
[4] W. Shang, Z. Jiang, H. Hemmati, B. Adams, A. Hassan, and P. Martin, "Assisting developers of big data analytics applications when deploying on hadoop clouds," in *ICSE'13: Proc. of the 35th International Conference on Software Engineering*, 2013, pp. 402–411.
[5] Q. Lin, H. Zhang, J. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in *ICSE'16: Proc. of the 38th International Conference on Software Engineering*, 2016.
[6] H. Mi, H. Wang, Y. Zhou, M. R. Lyu, and H. Cai, "Toward fine-grained, unsupervised, scalable performance diagnosis for production cloud computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, pp. 1245–1255, 2013.
[7] J. Zhu, P. He, Q. Fu, H. Zhang, M. R. Lyu, and D. Zhang, "Learning to log: Helping developers make informed logging decisions," in *ICSE'15*.
[8] P. He, J. Zhu, Z. Zheng, J. Xu, and M. R. Lyu, "Location-based hierarchical matrix factorization for web service recommendation," in *ICWS'14*, 2014.
[9] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "An evaluation study on log parsing and its use in log mining," in *DSN'16*.
[10] https://github.com/logpai/logparser.
[11] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in *ICWS'17: Proc. of the 24th International Conference on Web Services*, 2017.
[12] X. Yu, P. Joshi, J. Xu, G. Jin, H. Zhang, and G. Jiang, "Cloudseer workflow monitoring of cloud infrastructures via interleaved logs," in *ASPLOS'16: Proc. of the 21st International Conference on Architectural Support for Programming Languages and Operating Systems*, 2016.
[13] A. Nandi, A. Mandal, S. Atreja, G. B. Dasgupta, and S. Bhattacharya, "Anomaly detection using program control flow graph mining from execution logs," in *KDD'16: Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.
[14] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: System log analysis for anomaly detection," in *ISSRE'16: Proc. of the 27th International Symposium on Software Reliability Engineering*, 2016.
[15] https://github.com/cuhk-cse/loglizer.