An Analysis of Burstiness and Causality of System Logs

Kazuki Otomo[1], Satoru Kobayashi[1], Kensuke Fukuda[2], Hiroshi Esaki[1]
University of Tokyo[1], NII/Sokendai[2]
{otomo, sat}@hongo.wide.ad.jp, kensuke@nii.ac.jp, hiroshi@wide.ad.jp

ABSTRACT

System logs are important data to detect system faults and diagnose root causes of them in a large scale network system. However, due to a huge amount and wide diversity of logs, it is not easy and time consuming for network operators. This paper focuses on burstiness and causality of log time series data to extract meaningful information for troubleshooting. With Kleinberg's burst detection algorithm, we conduct three types of burstiness analysis depending on the combination of the log time series generated by 15 months syslog data obtained in an academic network in Japan: single, pair, and device-based burst detections. The contribution of this paper is as follows. In the single burst detection, we confirm our preprocessing can remove over 90% of trivial bursts. Next, in the pair burst analysis, we investigate causality of co-occurred bursts with causal inference results [9] and find that 99% of pair bursts are coincident; remaining 1% are causal pair bursts. Furthermore, our similarity analysis distinguishes two types of pair bursts depending on complexity of network event causality. In the device-based burst detection, we find 3,735 bursts that are only found by this multivariate analysis. In addition, we find some causal bursts missed in previous causal inference results. To combine these findings, we can extract meaningful log bursts from all the detected ones.

CCS CONCEPTS

• **Networks** → **Network manageability**; **Network management**;

KEYWORDS

Network log analysis; Burstiness; Causality

1 INTRODUCTION

Network reliability is a crucial concern in the Internet. For example, for ISP network operation, high availability and fault tolerance have been a critical requirement while its network system is getting more complicated and large-scaled. Therefore, network operators are required to troubleshoot network problems as fast as possible. Analyzing network logs is one of the most useful methods to understand natures of networks. In operational networks, syslog is widely used for collecting network logs and it allows us to gather logs from all devices at one server. With these logs, the operators investigate detailed status and events for each devices in the network system.

However, there are two issues to analyze system logs. First, a huge amount and kind of log messages from a large-scale network make the log analysis to be a hard task. Second, the important logs related to system faults are usually hidden in the majority of ones that report daily routine processes. Because of these two issues, manual inspection is an unrealistic method and automatic methods are highly required.

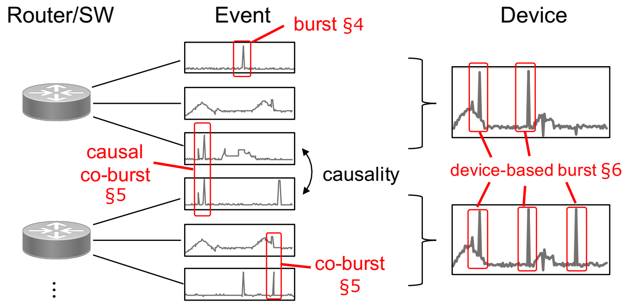


Figure 1: Analysis overview

In this paper, we overcome these problems in analyzing system logs to extract meaningful information for troubleshooting. We focus on log burstiness, which is one of the most popular time series characteristics, and its causality. In

AINTEC '17, November 20–22, 2017, Bangkok, Thailand

ACM ISBN 978-1-4503-5551-3/17/11…$15.00
https://doi.org/10.1145/3154970.3154973

other words, a question raised in the paper is how observed bursts in log messages are useful for network operation. To detect burstiness, we rely on Kleinberg's burst detection algorithm [8] and conduct three types of analysis depending on combination of the log time series: single, pair, and device-based burst detections (see also Fig. 1). We first apply the burst detection to the single log time series per event per day. After that, we investigate co-occurrence of bursts in a pair of such time series.We calculate burst co-occurrence ratio of pairs of bursts and compare with the causal inference results [9]. In device-based burst detection, we aggregate logs for each devices and apply burst detection. we focus on the bursts which can be detected only in this device level aggregation, but not in single event-based burst detection.

The contribution of this paper is as follows. First, our preprocessing removes over 90% of trivial bursts, in order to efficiently apply the Kleinberg's burst detection method to complicated and large scale network logs, Next, with three types of analysis and comparison with causal inference results, we extract meaningful burst data for troubleshooting from all the detected ones. Third, in device-based burst detection, we find some causal bursts which are not found in existing causal inference results. To combine with these results, we extract meaningful log bursts from a lot of trivial ones for troubleshooting and also extract causal relationships between log bursts.

2 RELATED WORKS

2.1 System log analysis

Many prior literature tackle on automated system log analyses. The automated system log analysis helps us detect system failures, identify the root cause of them, and predict the failures from tons of log messages. He et al. [5] discuss common functionality of the automated system log analysis by comparing prior literature: (1) log collection, (2) log parsing, (3) feature extraction, and (4) anomaly detection, root cause identification, or failure prediction. We conduct our analysis along with this classification. We need (1)-(3) processes to perform numerical or statistic methods to log messages because system logs are string data. In process (4), many analysis algorithms have been proposed.

To detect system anomaly with system logs, there are a lot of existing methods [14, 19]. In recent work, Baseman et al. [1] employ a graph analysis with a relational learning and kernel density estimation, and generate clusters of related syslog messages. They successfully extract anomalous behavior from super computer syslog data with low false positives. To diagnose root causes of system failure [4, 12] and predict system failure [15, 20] are also studied well. Lu et al. [11] propose a root cause analysis method with log messages and resource logs (e.g. CPU and memory logs). They focus on the task execution time and define features. To predict system failure, Kimura et al. [7] propose a network fault prediction system based on system logs. They analyze log features related to system faults in advance. With trouble ticket data, they incrementally analyze the log features and compare with the system fault log features.

ID	logs	devices	templates	filtering	bursts
Data 1	34.7M	130	1789	no	400,518
Data 2	2.3M	130	1789	yes	32,704

Table 1: SINET4 Dataset and analysis result

Overviewing these existing methods, we find there is an issue to be carefully considered; log anomaly do not directly indicate system anomaly. Many existing methods try to solve this issue with combining domain knowledge or sophisticated feature extraction. The goal of our work is to extract meaningful information for troubleshooting from system logs and our strategy is contextual log analysis with causal inference. The causal relationship of log anomaly relates to the real system anomaly although the simple co-occurrence of log anomaly includes a large number of false positives. Our method provides causality between groups of logs related to burstiness and the results are useful information for troubleshooting.

2.2 Causal inference from system logs

Causal inference is a statistical technique to identify a causal relationship between events. A well-known causal inference algorithm is PC algorithm [6, 16] that is based on conditional independence. The conditional independence distinguishes causal two events from co-occurred two events. Chen et al. [3] apply the PC algorithm to a set of time series (e.g., RTT, TCP window size) for identifying the source of network traffic delay.

The PC algorithm, however, has an issue for applying to system logs. Appearance of log messages is discrete and sparse compared to other metrics such as CPU or memory usage. It makes the causal analysis difficult. To overcome this issue, Kobayashi et al. [9] remove normal logs such as periodically or constantly appeared logs, and then apply the PC algorithm to an event time series to extract causal relationships. They apply their proposed algorithm to 15 months long syslog messages and successfully extract a small number of meaningful network events with causal relations. In this work, we use the same dataset and the causal inference results. We refer to a set of these results as the causality DB.

3 PRELIMINARY

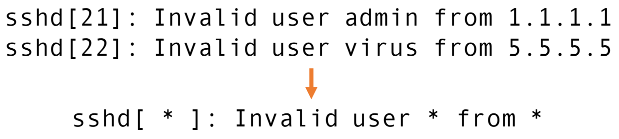
3.1 Dataset

We use a set of network logs collected at SINET [17], a Japanese research and education network. This network connects over 800 academic organizations in Japan and consists of eight core routers, 50 edge routers, and 100 layer-2 switches. We prepared two types of datasets; the original log messages (called Dataset 1) and the pre-processed log messages after removing frequent logs (called Dataset 2). We detail our preprocessing in §3.3. The summary of the datasets is shown in Table 1. We used a commodity computer for the analysis.

3.2 Log template generation

As log messages are string data, statistical approaches cannot be applied directly. Thus, we generate log templates from raw log messages and then we classify logs into log templates and extract time-series data from their time stamps for each template per device. The log template is a format of log messages composed of variables (e.g., IP address, port number) and other constants. Template generation problem is a well-known problem in log mining [13, 18]. In this work, we adopt a supervised learning approach proposed by Kobayashi et al. [10]. This algorithm is based on a CRF (conditional random field), which is well-studied in natural language processing and generates log templates composed by description words and variable words from original log messages. An example of log templates and raw log messages are shown in Fig. 2. The top two lines are raw log messages and the bottom line is a corresponding log template generated from them. "*" represents variable words. We manually fixed misclassified templates.

As the result of this processing, we generated 1,789 unique log templates. In this paper, we call the groups classified by each log templates as events.



```
sshd[21]: Invalid user admin from 1.1.1.1
sshd[22]: Invalid user virus from 5.5.5.5

          sshd[ * ]: Invalid user * from *
```

Figure 2: Example of log templates

3.3 Removing trivial log messages

The majority of log messages are related to daily processes, such as cron, ssh authentication, and so on. These logs are not helpful for trouble shooting because they commonly appear and indicate daily process results. Furthermore, to process a large amount of such trivial data cause serious accuracy degradation and time consumption. In particular, the causal inference method is greatly affected by frequent logs. Also, the burst detection method requires more time for processing larger data. To remove such the frequent and unimportant logs, we applied a frequent data filtering method to original log messages. This process has two parts: (1) Remove periodic logs by a Fourier analysis, and (2) Remove highly frequent logs by linear fitting appeared in very short period. In processing (1), we applied the Fourier transform to the time series, removed the data whose peak of frequency spectrum exceeded a threshold, and applied the inverse Fourier transform to reconstruct the time series. In Processing (2), we applied the linear regression to a cumulative time series of the data and calculated the regression error. If the error is below a threshold, we removed the data. We combined these two approaches because the Fourier analysis may miss very short periodic data and the linear regression analysis may not remove long interval data. Finally, we obtained non-periodical log time series from the original log messages.

3.4 Burst detection

We detect log burstiness from log message time-series. The log burstiness is a local state of a large number of log appearances. Kleinberg's burst detection algorithm [8] is a well-known method to detect burstiness from stream data. We apply this algorithm to network log time series for burst detection.

Kleinberg's burst detection algorithm simulates burstiness states for each time series data using an infinite-state weighted automaton model. This method estimates the burstiness states by minimizing a state transfer cost function. It is a batch processing and detects relative burstiness in the input data. We briefly review the algorithm. It evaluates burstiness based on the interval of the input data. The intensity of burstiness is given by burst level i. When the burst level is i in a state, the data arrival rate is distributed exponentially $f(x) = \alpha e^{-\alpha x}$. α is defined as $\alpha_i = \frac{n}{T} s^i$ where n is the number of input data and T is a time length of input data. s is a scaling parameter that indicates an arrival rate difference between neighboring states. Each automaton states have one burst level. State transition q is equal to maximizing the posteriori probability $P(q|x)$ where x is input data. Also maximizing $P(q|x)$ corresponds to minimizing a state transition cost function $c(q|x)$. $c(q|x)$ is represented as a sum of a threshold parameter γ and $p(x|q)$.

4 SINGLE BURST ANALYSIS

4.1 Methodology

We first analyze burstiness of single event time series with the burst detection algorithm. We divide the dataset to one-day long time series per event per device and detect burstiness in seconds. We implemented the detection algorithm by

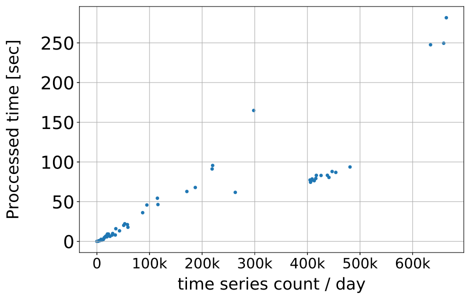


Figure 3: Processing time

Python 3.6 and Pybursts, which is a python module of burst detection. We empirically set the scaling parameter $s = 2.0$ and the threshold parameter $\gamma = 1.0$. As the algorithm does not support a case of multiple events at the same time, we randomly shift these appearance time to 0.001 seconds (i.e., serialized). We use the minimum level of the burstiness for our analysis while the algorithm outputs several levels of the burstiness.

We first demonstrate the processing time of the burst detection algorithm is reasonable. Figure 3 shows the number of time series in a day and its processing time. We confirm an increase in the processing time for the number of time series. All of one day data is processed within 300 seconds in the commodity hardware.

4.2 Result of burst detection

We apply the burst detection to each event of Dataset 1 and 2. As summarized in Table 1, we detected 400K bursts in the non-filtered data and 32K bursts in the filtered data. These correspond to about 800 and 70 bursts per day per event. In other words, only 7.5% of bursts are potentially meaningful. Figure 4 illustrates some examples of detected burstiness data. Each graph shows one-day cumulative time-series of an event. Gray rectangles indicate detected bursts.

We confirm mainly two types of burstiness; Burst with periodic data (Fig. 4(a)) and Pure burst (Fig. 4(c)). All of the bursts obtained in the dataset are not always useful for the network operation. For example, periodic burst due to daily processing are remained in Fig. 4(b). Indeed, this is a time series in Dataset 1 and this type of data is successfully removed in Dataset 2. These results demonstrate the importance of pre-processing, especially, removing the strictly periodic logs from the data. On the other hand, like Fig. 4(a) and Fig. 4(c), bursts with periodic data and pure bursts are important information for troubleshooting because changes of log appearances are a signal of system state changes.

5 PAIR BURST ANALYSIS

In this section, we intend to extract meaningful burst event pairs for troubleshooting from the results of the single burst detection using the co-occurrence of bursts and causality between events. First, we analyze the event co-occurrence related to burstiness using single burst detection results. Next, to analyze causality of burstiness, we compare the co-occurrence of bursts with the causal inference results. Finally, we classify causality of burstiness based on an event similarity using dynamic time warping distances [2].

5.1 Co-burst analysis

5.1.1 Methodology. We define Co-burst and a burst co-occurrence ratio to evaluate relevance between two bursts. Co-burst is a pair of two co-occurred bursts in two event time series. Co-burst event pair is a pair of two events which include at least one co-burst. We define the co-occurrence of two bursts if they start in the same 1-min bin. We calculate the co-occurrence ratio of events A and B as $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$. $J(A, B)$ is the Jaccard similarity coefficient [1]; The higher $J(A, B)$, the more frequent bursts occur at the same time between event A and B. A and B are events that are sets of time series data classified by each log templates, and $|A|$ and $|B|$ are the total number of bursts detected in 456 days. $|A \cap B|$ is the total number of co-bursts between event A and B in 456 days. $|A \cup B|$ is a disjunction of A and B.

5.1.2 Co-burst analysis result. We calculate the burst co-occurrence ratio using the single burst detection result (§4.2). The number of co-burst event pairs is 29,107 in Dataset 1 and 22,756 in Dataset 2. The distribution of the co-occurrence ratio is shown in Figs. 5 and 6. The vertical line shows the co-occurrence ratio in a log scale and the horizontal one the size of the union set of each event pair. The larger $|A \cup B|$, the more number of bursts is occurred in event A or B. Each dot shows a co-burst event pair.

In Dataset 1, 22% of dots are located in $|A \cup B| > 10,000$. These pairs are mostly periodic bursts, shown in Fig. 4 (b). As in §4.2, these bursts should be removed because they are the result of less useful daily processes. On the other hand, shown in Fig. 6, such larger points ($> 10,000$) disappeared in Dataset 2. This result confirms that the pre-processing in §3.3 works correctly and is effective to remove the trivial burst pairs.

5.2 Comparison with causality DB

Here, we compare the co-burst analysis results to the causality DB that includes 2,776 causal event pairs obtained in the prior work [9]. If the same event pair of co-burst events is recorded in the causality DB, we consider that the co-burst

[1] We also obtained similar results with the Simpson index.

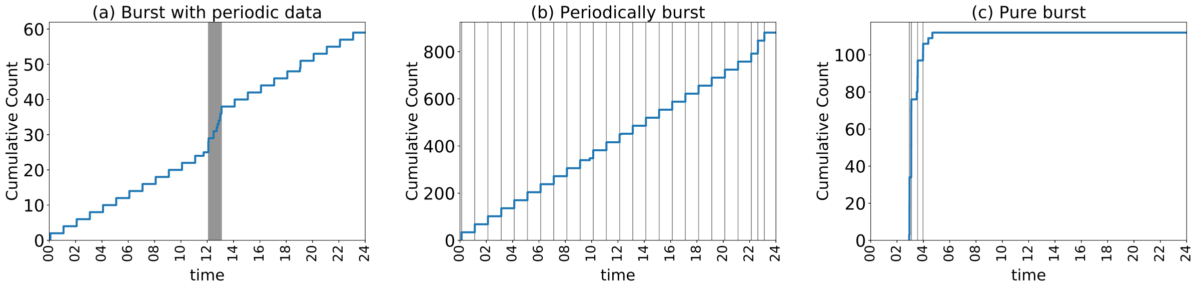


Figure 4: Burstiness event examples

	Co-burst	Causal Co-burst
all	22,756	241 (1.06 %)
$J(A,B) \geq 0.1$	17,541	215 (1.23 %)

Table 2: Co-burst and event causality

is not a coincident but a causal co-occurrence. We call such event pair as a causal co-burst. Table 2 shows the number of the causal co-bursts. The first row shows the total number of causal co-burst event pairs and the second one shows the number of events for $J > 0.1$. Red dots in Fig. 6 illustrate such causal co-burst events, while gray dots illustrate not causal but co-burst events.

Our results highlight two findings. First, we find a relationship between the causal co-burst events and the burst coefficient ratios. In Fig. 6, most red dots located at an upper part of the graphs, and over 89% of causal co-burst events have larger than 0.1 coefficient ratio from Table 2. Focusing on horizontally distributed causal co-bursts (red points), we can see two clusters; a small burst disjunction cluster ($|A \cup B| < 500$) and large one ($|A \cup B| > 500$). Inspecting the detail of the large one, there are 1316 pairs and we confirm 487 pairs are derived from "show interface" command (i.e., ordinal operation). The others (829 pairs) are triggered with this command by chance. On the contrary, there are only 1 pair related to the command in the small cluster. The small cluster has 21,440 pairs and the number of the causal co-burst pairs is 167. There are 156 causal co-burst pairs in this cluster. Thus, we conclude that the causal co-burst event pairs have relatively high coefficient ratios and the frequency of their bursts is comparatively low. Therefore, meaningful causal bursts are located at the upper left cluster of red points in Fig. 6. Second, we confirm the effectiveness to combine the causal inference results with the burst detection. Combining the causal inference, we observe only 1% of them are remained. Therefore, 99% of co-burst event pairs are coincident and we have to focus on 1% of them for extracting meaningful bursts.

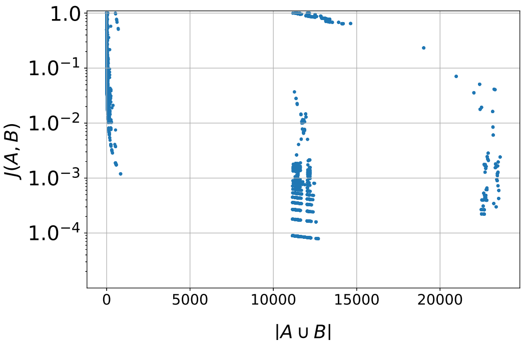


Figure 5: Burst co-occurrence ratio (Dataset 1)

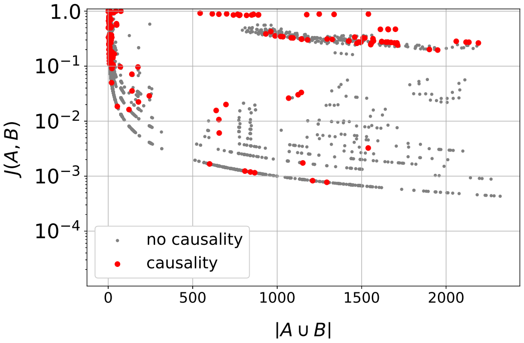


Figure 6: Burst co-occurrence ratio (Dataset 2)

5.3 Time series similarity and co-burst classification

Here, we measure the time series similarity between causal co-burst event pairs and classify the type of co-burst towards practical system management.

5.3.1 Time series similarity and causality. Figures 7 and 8 visualize examples of temporal patterns of causal co-burst. Orange and blue plots indicate two types of events. Checking the details of the log events, we confirm these examples

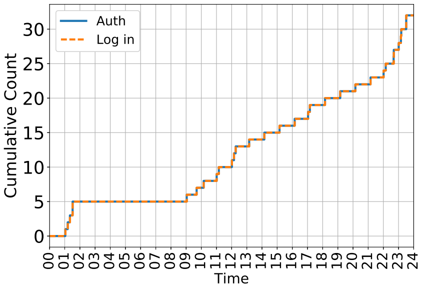


Figure 7: Co-burst (from single device)

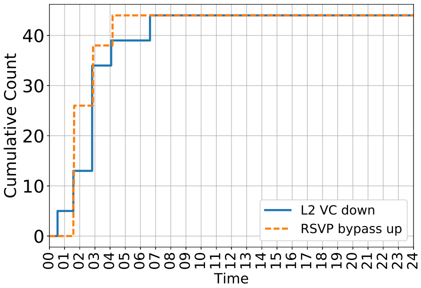


Figure 8: Co-burst (from multiple devices)

are helpful in network operation. In Fig. 7, login and authentication events in one device obviously have a causal relationship. Also in Fig. 8, an outage of a L2 switch invokes a bypass event, which means a change of network routes to avoid outage devices and keep network availability.

However, these two examples are different in whether they are triggered from one network activity or multiple ones. As shown in Fig. 7, log in activity always includes both log in and authentication event. On the other hand, in Fig. 8, a system down and enabling bypass events are not always appear at the same time. This is because system down and bypass event come from different network activity. Therefore, there are two types of event causality. (1) Causality from one network activity, and (2) Causality between different network activities. This difference also appears in time-series. Two events in the pair of type (2) exhibit almost the same pattern during a whole time-series, while the type (1) only shows the same pattern locally. Thus it is valuable to classify these two types of co-burst based on a similarity of whole time-series because we can distinguish whether causal co-burst is type (1) or type (2).

5.3.2 Time series similarity analysis with Dynamic Time Warping distance. We introduce a similarity of temporal patterns of causal co-burst events by using Dynamic Time Warping (DTW). This similarity enables us to distinguish whether causal co-burst events come from different network activities or same one.

To calculate the similarity of two events, we should consider a case that two time series are mostly similar but partially different. Thus, the strict comparison of two time series may miss potentially similar events. DTW is a dynamic scheduling method to compare the similarity between two time-series. It can disregard the difference of periodicity and data size. First, DTW processes data points matching between two events which minimize distance of matching points. We choose the absolute distance of log occurrence time as an inter-points distance. Then, we aggregate all the inter-points distance as the DTW distance (similarity).

We apply this method to all the causal co-burst event pairs per day. We set the similarity threshold to 10,000, meaning that we permit up to 10,000 seconds-length (i.e., about 3 hours) errors in total per day. For each event pairs, if DTW distance is under threshold, we classify them as similar pairs and otherwise as dissimilar ones. The result is shown in Fig. 9. The horizontal line indicates manual classification of events and vertical one does the number of appeared days. In the each bar, blue part shows similar pairs and orange part shows dissimilar ones. There are 40 pairs of event classes and 13 pairs are classified dissimilar in more than half of appeared days (red labels). To focus on these 13 pairs, they include 9 pairs of two different classes. We consider these types of pairs as type (1) (see §5.3.1) and we successfully classify them with DTW distance. Therefore, we can distinguish whether a causal co-burst comes from type (1) or type (2) based on time series similarity. This metrics is useful to evaluate the complexity of causality.

6 DEVICE-BASED BURST DETECTION

Previous sections focused on the burstiness appeared in time series per event per device. Here, we analyze time series aggregating all events per device.

6.1 Device-based log burstiness

We aggregate events per device and apply the burst detection to them; The detection method and its parameter setting are the same with the single burst detection. There are 130 devices and 1,789 templates in Dataset 2. We aggregate events across all the templates in each devices (i.e., 130 devices × 456 days). We detect 21,670 bursts and refer to them as device-based bursts.

We find two types of device-based burst by analyzing major templates contributing to device-based bursts. One is the burst that is also detected in each events (§4.2) and the other is only detected in this device-based analysis. Figure 10 (a), (b) and (c) shows an example of the device-based burst.

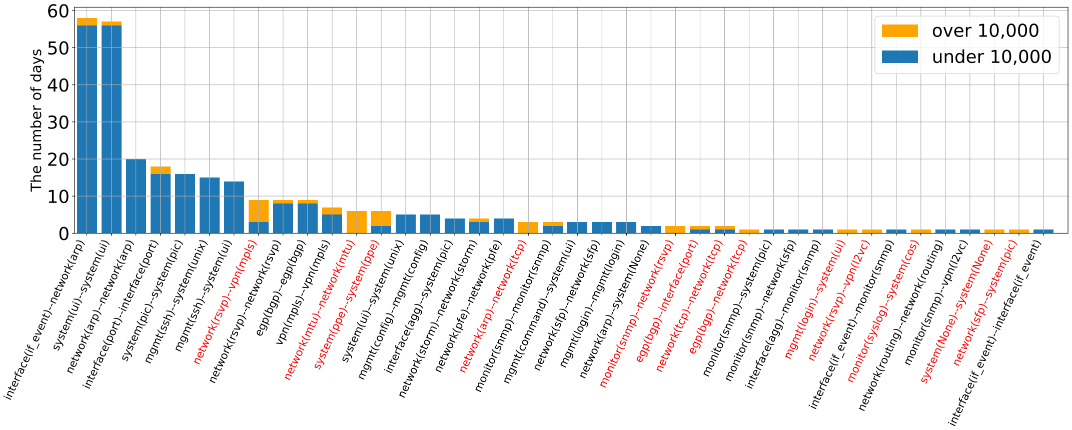


Figure 9: DTW distance distribution. Red labels show dissimilar pairs classified by DTW distance.

The black line shows device-based time-series. Other lines show some event time series in the device.

Commonly, one or two events dominate a majority of the device-based logs in Fig. 10(a). Thus, some device-based bursts appear due to a few dominant events and these bursts have been similarly detected in the single burst analysis (§4.2). We can see that a burst starts in two major events (events 1 and 2) as well as the device-based burst at 3:00. On the contrary, a device-based burst also appears when many types of small events occur at the same time. In Fig. 10 (b), we can see a device-based burst at 20:00 without such majorities. We further focus on the latter type of bursts (such as shown in Fig. 10 (b) and (c)) because the previous event-based analysis missed finding them.

6.2 Device-based burst detection

We extract the bursts that do not appear in the event-based analysis from the device-based bursts. Here, we focus on such 3,735 bursts extracted from 21,670 bursts.

To investigate events contributing device-based bursts, we dig into what happened at that time. Figure 10 (b) shows an example of a device-based burst. Observing many events related to system standby and power supply occurred at the same time, we understand a system reboot happened on this device due to some reason.

In another example (Fig. 10 (c)), a device-based burst appears from 22:00 to 23:00 as well as many OSPF and BGP events. This suggests some routing errors. Note that this burst cannot be detected by the event-based analysis, thus, the device-level aggregation is useful as well as the event-level analysis.

6.3 Comparison with causality DB

To analyze the causality between events in device-based bursts, we compare them with the causality DB. Because we only know causality between two events, not multiple events from the causality DB, we inspect whether the event pairs have causality. Figure 11 shows the ratio of the number of causality event pairs to all the pairs of events contributing device-based bursts. The vertical axis indicates the ratio: $\frac{\text{CausalEventPairs}}{{}_{Events}\text{C}_2}$. The horizontal line shows the number of events and each point corresponds to device-based bursts. We confirm that most events composing device-based bursts do not have causality. However, we find some cases in which events in the device-based burst have causality considering from log contents, not from the causality DB. Figure 10 (c) is such an example. A device-based burst is appeared in around 22:00-24:00 and both OSPF and BGP events contribute to the device-based burst. The two events obviously have causality in terms of network operation. However between BGP and OSPF events, time-series causality is not recorded in the causality DB. Therefore device-based burst analysis can detect causality which cannot be detected by time-series causal inference method and device-based burst analysis can improve the causality analysis.

7 CONCLUSION AND FUTURE WORK

We focused on burstiness and causality appeared in network log messages. We conduct three types of analysis: single, pair, and device-based burst detection. To combine burst detection results, causal inference results and other metrics such as burst co-occurrence ratio and dynamic time warping distance, we reduce a large amount of trivial bursts, and extract meaningful information from log messages. We find

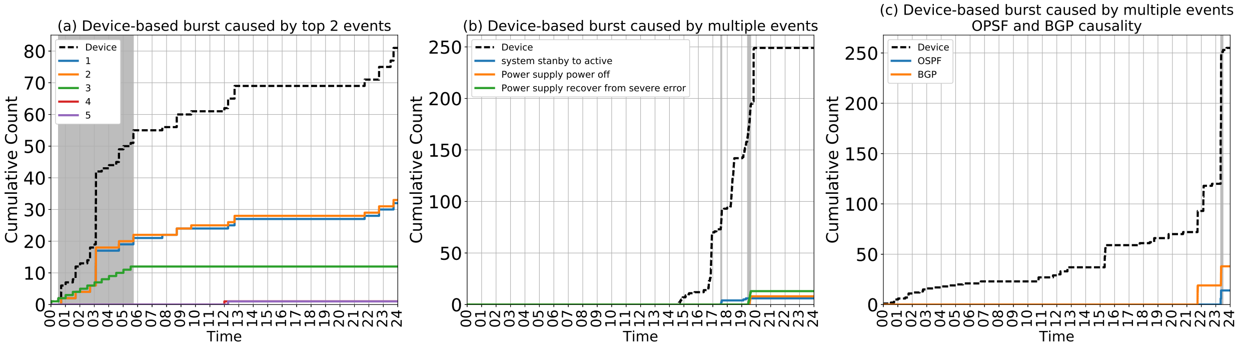


Figure 10: Device-based burst examples

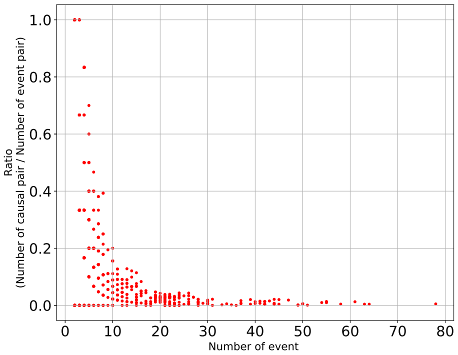


Figure 11: Device-based burst distribution

that 7.5% of co-occurred bursts have true causality. Our similarity analysis separates whether the causality of co-burst comes from one network activity or multiple ones. In the device-based burst detection, we find 3,735 bursts that are only found by this multi variate analysis. Also we find some causal bursts which are not found in the existing causal inference results. As a future work, we plan to extend our results in this study to an automated log analysis system.

REFERENCES

[1] E. Baseman, S. Blanchard, and E. Zongzelimyuntedu. Relational Synthesis of Text and Numeric Data for Anomaly Detection on Computing System Logs. *in Proc. IEEE ICMLA'16*, 1:2–5, 2016.

[2] D. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. *in Proc. ACM KDD'94*, 398:359–370, 1994.

[3] P. Chen, Y. Qi, P. Zheng, and D. Hou. CauseInfer: Automatic and distributed performance diagnosis with hierarchical causality graph in large distributed systems. *in Proc. IEEE INFOCOM'14*, pages 1887–1895, 2014.

[4] 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. HiPC'10*, pages 1–10, 2010.

[5] S. He, J. Zhu, P. He, and M. R. Lyu. Experience Report : System Log Analysis for Anomaly Detection. *in Proc. IEEE ISSRE'16*, pages 207–218, 2016.

[6] M. Kalisch and P. Buehlmann. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *Journal of Machine Learning Research*, 8:613–636, 2005.

[7] T. Kimura, A. Watanabe, T. Toyono, and K. Ishibashi. Proactive Failure Detection Learning Generation Patterns of Large-scale Network Logs. *in Proc. CNSM'15*, pages 8–14, 2015.

[8] J. Kleinberg. Bursty and Hierarchichal structure in streams. *Data Mining and Knowledge Discovery*, 7(4):373–397, 2003.

[9] S. Kobayashi. Mining causes of network events in log data with causal inference. *in Proc. IEEE IM'17*, pages 45–53, 2017.

[10] S. Kobayashi, K. Fukuda, and H. Esaki. Towards an NLP-based log template generation algorithm for system log analysis. *in Proc. CFI'14*, pages 1–4, 2014.

[11] S. Lu, B. Rao, X. Wei, B. Tak, L. Wang, and L. Wang. Log-based Abnormal Task Detection and Root Cause Analysis for Spark. *In Proc. IEEE ICWS*, 2017.

[12] A. Makanju, a. N. Zincir-Heywood, and E. E. Milios. Storage and retrieval of system log events using a structured schema based on message type transformation. *in Proc. ACM SAC'11*, pages 528–533, 2011.

[13] M. Mizutani. Incremental mining of system log format. *in Proc. SCC'13*, pages 595–602, 2013.

[14] M. Moh, S. Pininti, S. Doddapaneni, and T.-S. Moh. Detecting Web Attacks Using Multi-stage Log Analysis. *in Proc. IEEE IACC'16*, pages 733–738, 2016.

[15] M. Shatnawi and M. Hefeeda. Real-time failure prediction in online services. *in Proc. IEEE INFOCOM'15*, pages 1391–1399, 2015.

[16] P. Spirtes and C. Glymour. An Algorithm for Fast Recovery of Sparse Causal Graphs. *Social Science Computer Review*, 9(1):62–72, 1991.

[17] S. Urushidani, M. Aoki, K. Fukuda, S. Abe, M. Nakamura, M. Koibuchi, Y. Ji, and S. Yamada. Highly available network design and resource management of SINET4. *Telecommunication Systems*, 56(1):33–47, 2014.

[18] R. Vaarandi. A data clustering algorithm for mining patterns from event logs. *in Proc. IPOM'03*, pages 119–126, 2003.

[19] W. Xu, L. Huang, A. Fox, D. Patterson, M. I. Jordan, L. Huang, A. Fox, D. Patterson, and M. I. Jordan. Detecting Large-Scale System Problems by Mining Console Logs. *In Proc. ACM SIGOPS'09*, pages 117–131, 2009.

[20] J. Zhong, W. Guo, and Z. Wang. Study on network failure prediction based on alarm logs. *2016 3rd MEC International Conference on Big Data and Smart City, ICBDSC 2016*, (2015):23–29, 2016.