A Hierarchical Tree-Based Syslog Clustering Scheme for Network Diagnosis

Sizhe Rao*†, Minghui Wang*†, Cuixia Tian†, Xin'an Yang†, Xiangqiao Ao†
†AI Institute of H3C Technologies Co., Ltd., Beijing, China
Email:{rao.sizhe, mhwang, tian.cuixia, yangxinan, aoxiangqiao}@h3c.com
*Equal contribution.

***Abstract*—With the continuous development of Information Technology, modern networks have been widely utilised. Since the complex network structure causes growing difficulties in maintenance, log analysis has been widely studied in recent years for network diagnosis. System log clustering is mainly focused for root cause analysis. In this paper, a hierarchical tree-based clustering scheme is proposed that could accurately group system logs according to both time and network constraints without any training and parameter settings. Furthermore, it largely accelerates the matching process by reducing matching times and significantly boosts the performance of hit rate (100%) and match efficiency (16%) comparing to other clustering strategies, which greatly helps with precise network diagnosis.**

***Index Terms*—system log clustering, hierarchical tree-based clustering, correlation rule matching, network diagnosis**

I. Introduction

With the continuous development of Information Technology, modern networks have been widely utilised. As the applications are supposed to cover larger area, the amount of devices are keeping increasing to form a more complex network structure which causes growing difficulties in maintenance for operation staff. A large amount of system logs (syslog) are produced everyday with different severity levels such as notification, warning, error, alert etc. Among these syslog, faults need to be successfully recognised and corresponding root causes need to be correctly targeted to mitigate the impacts of recognised faults. However, these maintaining processes are exceedingly time-consuming and currently require heavy manual efforts from operation experts. For instance, a single fault might cause hundreds or thousands of syslog in very short time which makes real-time maintenance almost impossible [1], [2]. Consequently, a reliable and stable system that could efficiently achieve real-time self-diagnosis and mitigate the incident is in high demand.

The self-diagnose process could be regarded as a sequential one including symptom extraction and actual diagnostic task [3]. In symptom extraction part, features are extracted from the observations. In other words, the system needs to understand what happened in the networks according to given syslog and detect faults in time. Then the actual diagnostic task would localise root causes for the faults by the built diagnose engine and notify the operation staff to check the corresponding part such as physical connections or network configurations etc. Considering this, it is essential to effectively extract symptoms from syslog to ensure the accuracy in actual diagnostic task.

In recent years, log analysis has been widely studied to achieve effectively symptom extraction. Real-time records from all devices in the networks are saved to the log system, including their running status, operations, warning messages etc. Automatic log analysis could explore these raw data by template extraction [4]–[6], clustering [7]–[9] and anomaly detection [10], [11]. Given merged clusters from above log analysis processes, related fault information could be obtained from a large amount of syslog by correlation rule matching.

Correlation rules are often acquired from classic association rule mining algorithms [12]–[14]. Mining rules within clusters could help to merge related syslog together and get the most informative ones. However, manual verifications for mined rules are also necessary by checking their correctness and adding specific satisfying requirements to the rules aiming to ensure the reliability. Given accurately clustered warning groups, the rule matching engine could provide precise syslog correlations which automatically assists fault targeting. In consequence, network diagnosis could be performed with much less human efforts which significantly accelerates network diagnosis comparing to manual maintenance by operation staff.

In this paper, a hierarchical tree-based syslog clustering scheme is proposed that could accurately group the syslog according to both time and network constraints. Our approach successfully achieves better performance for syslog clustering in practical use with less human efforts and mainly contributes in two aspects:

1) Better performance without any training and parameter settings. Considering strict data limitation and complex model training in previous studies, we provide a training-free and parameter-free clustering scheme that could be easily adapted to real-world circumstances.
2) Specific fault targeting. From previous studies, experts need to diagnose the fault based on given IP address, summarised sequences or general suggestions. However, it would still be long time for them to find out the detailed problem on a specific device and solve it. Regarding our tree-based method, preserved hierarchical information could help us to target specific problems on a device (e.g. slot 1 on device A) which allows experts to quickly and accurately solve the problem.

978-3-903176-36-2 ©2021 IFIP

II. Related Work

Template extraction is required to reformat and normalise the unstructured syslog due to different device types and log formats. Two basic solutions commonly used are static and dynamic ones [15]. The static method usually parse the syslog according to known templates or codes which indicates high accuracy and speed. However, it requires prior knowledge and shows weak generalisation capacity on unknown devices and log formats [10], [16]. While the dynamic method basically apply unsupervised approaches to automatically learn the templates by frequent pattern mining [17], [18] or clustering [4], [19], followed by template matching for given syslog.

Despite the high generalisation capacity for dynamic approaches, only static parsing will be employed in this paper since all devices we included are H3C switches with fixed log templates. Moreover, although dynamic approaches could automatically obtain the semantic information from syslog, those information are regarded as independent variables for further clustering without involving their internal correlations. Consequently, the dynamic approaches could barely recognise the hierarchical structure of the network elements without post-processing under human-defined rules which is the prerequisite for network constraints in our tree-based clustering scheme.

Syslog clustering is conducted aiming to group related syslog and obtain significant fault information. The syslog relationships could be represented by textual similarity, topological/network correlation and time correlation. Clustering algorithms including partition-based K-means [20] and density-based DBSCAN [21] are commonly used to group syslog according to desired correlations. Apart from these, [7] proposed Agglomerative Hierarchical Clustering strategy to merge clusters from single log without prior knowledge for cluster amount and density estimation.

Textual similarity for syslog could be estimated by approaches in Natural Language Processing which are simply based on the log contents excluding device settings or topological information [22]. [23] introduced a cell-based clustering algorithm that calculates Euclidean Distance between extracted features of syslog (CPU, memory, I/O and network utilisation). However, since most syslog contain not only those numbers but also names of network elements and warning messages, clustering based on Edit Distance [24] and Jaccard Distance [8] between log sentences are more appropriate to evaluate textual similarity.

Nevertheless, simple textual similarity could not fully represent the relationship among logs. Our devices in modern networks are not isolated and they are related to each other through physical or logical links which form the whole topological network. Moreover, syslog are time-sensitive and they may not be correlated if large time intervals exist in between. Considering both topological and time constraints, correlations on these two aspects would probably be more important than textual similarity.

Regarding topological and time correlation, [2] and [9] conducted similar methods to cluster syslog based on network and time constraints. [2] divided the whole topological network by different devices to ensure syslog in one group are all belonging to the same device or network element. Moreover, DBSCAN algorithm is applied to further divide the obtained group according to time density. While [9] defined three types of topological relationships including syslog from the same network element or device, syslog in the same transmission section and syslog located in the same channel. Time correlation is constraint by pre-defined time window since syslog are regarded as related ones only when they occur in a short period of time.

Furthermore, [8] raised a more advanced clustering algorithm to combine all three types of correlations. In their work, similarity matrix for logs in one minute is created by combining both textual and topological similarity. Jaccard Distance is employed to calculate the word differences between logs ignoring the position information while directed graphs for both service and server are generated to find paths between logs to estimate the topological distance. Two similarity values are merged by a weighted equation to get the final similarity in the matrix whose weights could be alternatively modified according to real circumstance by operation experts. In addition, DBSCAN algorithm is used to find the density regions and expand the cluster according to the similarity matrix. Before the clustering process, density threshold and minimum number of syslog in a cluster are supposed to be specified.

Although above work could successfully achieve real-time clustering when dealing with large amount of syslog, it still cannot guarantee the entire coverage. The clustering methods have lowest limitation for syslog amount in one cluster which means if a log or several logs are much less similar than others, they are very likely to be filtered during the clustering process as outliers. To address this weakness, our hierarchical tree-based clustering approach preserves the real-time clustering performance under network and time correlation constraints with no limitation for cluster size. In this way, the subsequent rule engine could perform full correlation matching based on entire syslog (without missing less-frequent but necessary syslog) which ensures the accuracy in further network diagnosis.

III. Methodology

The workflow of our entire intelligent alarming system is described in Fig. 1 in order to target root causes for specific faults based on syslog. In this paper, we mainly focus on clustering module aiming to provide precise clustering results for correlation rule matching. Details of our hierarchical tree-based clustering scheme will be described in this section including pre-processing and clustering as two main steps.

A. Syslog pre-processing

In the pre-processing part, syslog will be collected from Kafka and parsed by static template matching. Then the parsed structured logs will be filtered at the beginning according to their severity level and occurrence frequency to eliminate unnecessary ones, such as debugging information.

Fig. 1. Workflow of our entire intelligent alarming system.


```
{
 'timestamp': '2020-06-16T22:19:44+08:00',
 'message': '<4>Jun 16 22:19:44 2020 SXXXXE
      %%10IFNET/5/LINK_UPDOWN: Line protocol state on the
      interface Tunnel4 changed to up.',
 'host':  '2.2.2.2',
  'pri': 4,
  'logTime': 1592317184.0,
  'loghostname': 'SXXXXE',
  'module': 'IFNET',
  'severity': 5,
  'logTypeDesc': 'LINK_UPDOWN',
  'desc': 'Line protocol state on the interface Tunnel4 changed to up.',
  'ldp_uuid': '3252ca47-d94a-4d1a-b594-24f31ca82323',
  'NE': ('device=2.2.2.2', 'interface=Tunnel4'),
  'parameters': {'status': 'up'}
}
```


Fig. 2. Syslog parsed by Logstash.

1) syslog parsing: Aiming to make full use of syslog, semantic information need to be extracted from raw logs according to the static templates (H3C switch). As the example shown in Fig. 2, the obtained raw syslog (in the red box) is parsed to a dictionary-format data structure with general information (e.g. module, severity, log type), Network Elements (NE) representing its hierarchical structure within a device and parameters representing its detailed information (e.g. status, neighbouring address, session name).

Regarding different types of syslog, their parameters might be different. For instance, some syslog from ports have their current status but do not contain OSPF IDs that are usually included in some protocol syslog. Considering this, different syslog templates are statically formatted by Logstash [25] and template matching is operated according to different syslog types. In this way, detailed variables in specific positions could be matched with the templates and extracted to form the final parsed structures.

2) syslog filtering: Raw syslog from the collector are not always valid ones due to their incompleteness. When there are some errors on the slot, the syslog collected might have 'NULL' as its module name which is not valid for further clustering. Therefore incomplete syslog are filtered after checking the 'NULL' values in parsed logs.

Moreover, noise, such as logs with low severities and logs flapping in short time, will be removed as unnecessary syslog. For instance, DEBUG or INFO syslog from H3C switches have low severity level. Those syslog represent normal operating logs instead of warnings which are not useful in later network diagnosis. In addition, duplicated logs might continuously occur and represent exactly the same information which could be merged to a single log.

After above filtering, we could ensure the validity, correctness and effectiveness of the parsed logs. Then the operation staff could focus more on de-noised syslog which improves the efficiency of network diagnosis.

B. Syslog clustering

In this part, we proposed a hierarchical tree-based syslog clustering scheme which could involve both topological and time constraints to assist further correlation rule matching and network diagnosis. The clustering process could be divided into two parts: building a warning forest and merging time-correlated groups.

In warning forest building process, parsed syslog are initially sliced to sub-sequences by a 1-minute time window. Syslog within the same time window and from the same device will form a warning tree. Several trees could build the warning forest of this time window. Consequently, warnings with correlated occurrence time and topology are primarily gathered together.

Then, further clustering is performed based on above groups. Each warning will be iterated and inserted to the corresponding tree node (described in Algo. 1). The value of NE in each warning (see Fig. 2 for details) is actually an ordinal sequence representing the hierarchical structure of its network element. Therefore, by iterating items in NE, we could construct a forest for all warnings in the same time window.

Algorithm 1 Insert to Forest

```
Input: Warning to be inserted: warn;
       Tree-based warning forest: forest;

obtain NE of the warning: NE = warn.NE
initialise the curNode: curNode = forest.root
for item in NE do
    refresh time range of curNode according to warn
    if item is a one of the son nodes of curNode then
        curNode = son node of curNode
    else
        create a new son node of curNode: newNode
        curNode = newNode
    end if
end for
mark as an entity: curNode.isEntity = True
add the warning: addWarnList(curNode, warn)
```

The obtained warning forest is presented in Fig. 3 where the blue circles represent entities and white circles represent

dummy nodes. The warnings related to tree nodes are also displayed with the corresponding time range of each node. Then a bottom-up hierarchical clustering process (described in Algo. 2) is operated where warnings in each leaf of the forest form an individual warning group initially. Furthermore, the warning groups will absorb the warnings of their parent nodes if those are also entities (Fig. 4). Finally, after grouping warnings according to topological constraints, time correlation will be estimate by calculating time gap between two different groups. If their gaps are shorter than 10 seconds (log occurrence time difference between the last warning in one group and the first one in the other group), two groups will be merged as one for each independent warning tree (Fig. 5). Consequently, we could get syslog clusters under both topological and time constraints.

Algorithm 2 Bottom-up Hierarchical Clustering

```
Input: Tree-based warning forest: forest;
Output: Merged groups: m_groups;

get individual groups from each node: groups
sort groups by its first log occurrence time in each item
initialise merged group list :
      m_groups = [groups[0]]
initialise current index: idx = 0
pop the first item from groups, only iterate the others
for group in groups do
    if overlap between group and m_groups[idx] <10 then
        idx = idx + 1
    end if
    add warnings in group to m_groups[idx]
end for
```

IV. Experiments

In this section, details for experiments will be displayed including datasets, evaluation methods and corresponding experiment results.

A. Data

Here we employed clustering methods on two datasets including a smaller one collected from our local testing networks and a larger one collected from real-world scenarios.

The one from our local testing networks consists of around 400 valid syslog and covers all the correlation rules in the rule engine. Apart from these, around 330,000 real-world syslog from H3C switches were also collected over 15 days where 137,118 valid syslog form our real-world in-house dataset to evaluate our method and make comparison with two previous studies [2], [9]. Sample syslog fragment is displayed in Fig. 6.

In addition, correlation rules were mined by FP-Growth algorithm [26] with minimum support 0.2 and confidence level 0.8. After verified by experts, 60 in-house correlation rules are obtained for further evaluation.

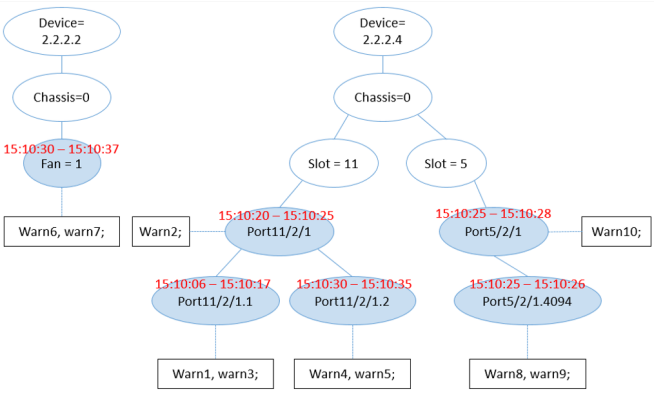

Fig. 3. Obtaining warning forest.

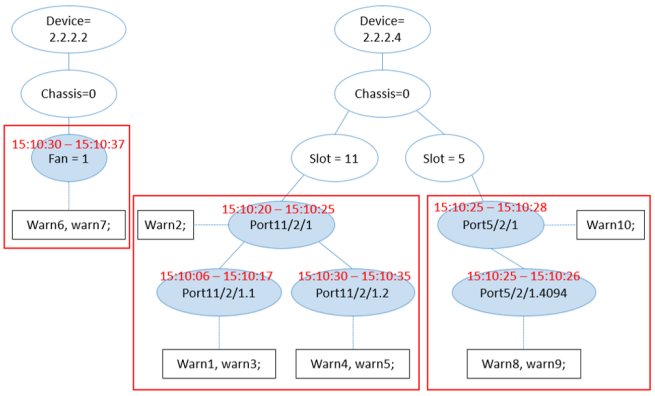

Fig. 4. Grouping tree nodes according to topology.

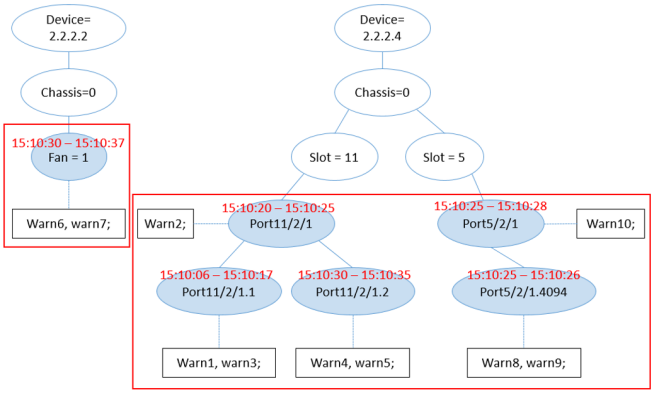

Fig. 5. Merging groups according to time overlap.

B. Evaluation

Evaluation is performed on our hierarchical tree-based clustering method and two previous studies [2], [8] (see details in Table I). The grouped syslog by three approaches are fed into the same rule engine to ensure a more precise comparison.

TABLE I
Comparison for three methods.

method	textual similarity	topological similarity	time correlation
[2]	N/A	device IP only	DBSCAN
[8]	Jaccard Distance	device topological distance	fixed time window
Ours	N/A	hierarchical structure	fixed time window

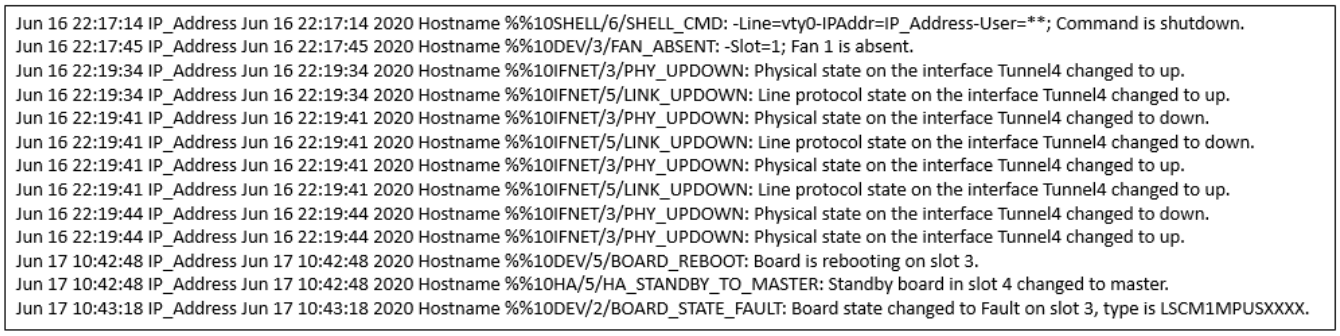

Fig. 6. Sample syslog from H3C switches.

As mentioned in Section II, [2] clustered the syslog by device IPs and used DBSCAN for time constraints while [9] clustered the syslog by device IPs, transmission sections and channels apart from DBSCAN used for time constraints. Considering that our syslog are all from the same transmission section and channel, constraints in [2] and [9] are almost the same which is unnecessary to evaluate on both of them. As for DBSCAN algorithm in [2], we set minPts equals to 2 and ϵ equals to 30 according to elbow method [21] which allow the approach to cluster syslog with time gap smaller than 30 seconds.

Moreover, to adapt [8] to our work, textual similarity by Jaccard Distance and server(namely device for our data) topological similarity are employed while service topological distance estimation is eliminated since no software services on the server side are involved in our project. In this way, the total similarity between each pair of syslog is:

$$\begin{aligned} Text(s_1, s_2) &= 1 - \frac{|bow(s_1) \cap bow(s_2)|}{|bow(s_1) \cup bow(s_2)|} \\ Topo(s_1, s_2) &= path_{server}(s_1, s_2) \\ Similarity(s_1, s_2) &= \alpha \times Text(s_1, s_2) + (1 - \alpha) \times Topo(s_1, s_2) \end{aligned} \tag{1}$$

where $bow(s_1)$ represents the bag of words of warning s_1 and $path_{server}(s1, s2)$ represents the shortest path length between two devices on topological graph. The weighted value α for similarity calculation is set to be 0.6 according to [8]. In addition, parameters of DBSCAN for similarity clustering are set to be minPts=2 and ϵ=0.8 according to elbow method [21].

Hit rate, match efficiency and real-time capability in correlation rule matching are the three main criteria for both accuracy and efficiency evaluation on all three clustering approaches:

$$\begin{aligned} \text{Hit rate} &= \frac{\text{hit amount}}{\text{desired hit amount}} \times 100\% \\ \text{Match efficiency} &= \frac{\text{hit amount}}{\text{total match amount}} \times 100\% \end{aligned} \tag{2}$$

where the hit amount represents the amount of successful matchings (all requirements satisfied for a rule), desired hit amount represents the amount of total correct matchings and total match amount represents the amount of the performed matching times for each algorithm during the correlation rule matching process.

Moreover, the desired hit amount is 204 and 73,237 for the datasets in local testing networks and real-world scenarios respectively. The desired hit amounts obtained by exhaustive search are regarded as truth for our evaluation process. We did rule matching between each pair of syslog with each rule and calculate the total matched amount for two datasets. It is a time-consuming but accurate process that requires no human effort for validation and could ensure 100% coverage of correct matchings. Based on the truth, we could perform convincing comparison for the three approaches.

C. Results

The evaluation results for above three approaches on two prepared datasets are displayed in Fig. 7. It is clear that for both datasets, our hierarchical clustering method and [2] could achieved 100% hit rate in correlation rule matching which guarantee the precision in further network diagnosis. Moreover, our method achieved the highest match efficiency which could effectively limit the resource usage to ensure the system efficiency.

Regarding the low hit rate in [8], a cropped syslog fragment is used to make the comparison. For instance, syslog in Fig 8 represent one fault with the following correlation order: 'BOARD_REBOOT' $\rightarrow$ 'BOARD_STATE_FAULT' $\rightarrow$ 'PHY_UPDOWN' $\rightarrow$'LINK_UPDOWN'$\rightarrow$'OSPF_NBR_CHG'. The fault contains status syslog from specific board, port and protocol.

Grouped results are displayed in Fig 8 in which syslog in the same block represent the same cluster. It is obviously that our clustering method grouped all 5 records into the same group due to the time and topological constraints which could be used for network diagnosis afterwards. However, these syslog are separated into three different clusters according to [8] since textual dissimilarity breaks the inner-correlation.

However, in most scenarios, correlated syslog do not share similar textual structure which might be a defect for involving this constraint. Considering this, we tested for different ϵ values for DBSCAN in [8] from 0.8 to 0.5 and the results showed that lower ϵ could fairly increase the hit rate but the match efficiency significantly dropped. This whole clustering process changed gradually towards exhaustive search in one-minute time window. Namely, [8] with lower ϵ value (lower similarity threshold) becomes the real-time work of [2].

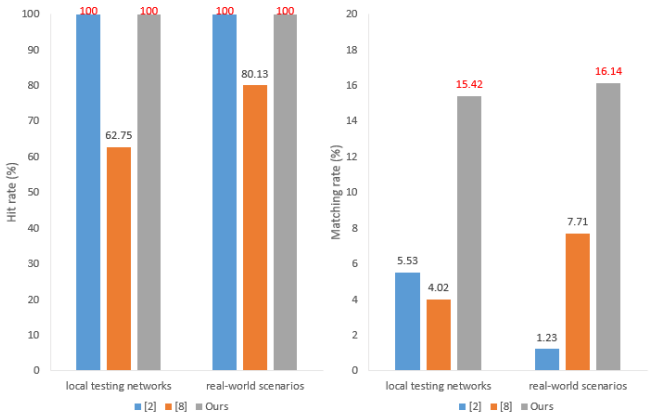

Fig. 7. Correlation rule matching results for three methods.

Clustered by our method	Clustered by [8]
Jun 16 22:21:43 IP_Address Jun 16 22:21:43 2020 Hostname %%10DEV/2/BOARD_STATE_FAULT: -MDC=1; Board state changed to Fault on slot 2, type is LSXM1CGQ36TD1.	Jun 16 22:21:43 IP_Address Jun 16 22:21:43 2020 Hostname %%10DEV/2/BOARD_STATE_FAULT: -MDC=1; Board state changed to Fault on slot 2, type is LSXM1CGQ36TD1.
Jun 16 22:21:52 IP_Address Jun 16 22:21:52 2020 Hostname %%10DEV/5/BOARD_REBOOT: -MDC=1; Board is rebooting on slot 2.	Jun 16 22:21:52 IP_Address Jun 16 22:21:52 2020 Hostname %%10DEV/5/BOARD_REBOOT: -MDC=1; Board is rebooting on slot 2.
Jun 16 22:21:43 IP_Address Jun 16 22:21:43 2020 Hostname %%10IFNET/5/LINK_UPDOWN: -MDC=1; Line protocol state on the interface Vlan-interface4094 changed to down.	Jun 16 22:21:43 IP_Address Jun 16 22:21:43 2020 Hostname %%10IFNET/5/LINK_UPDOWN: -MDC=1; Line protocol state on the interface Vlan-interface4094 changed to down.
Jun 16 22:21:43 IP_Address Jun 16 22:21:43 2020 Hostname %%10IFNET/3/PHY_UPDOWN: -MDC=1; Physical state on the interface Vlan-interface4094 changed to down.	Jun 16 22:21:43 IP_Address Jun 16 22:21:43 2020 Hostname %%10IFNET/3/PHY_UPDOWN: -MDC=1; Physical state on the interface Vlan-interface4094 changed to down.
Jun 16 22:21:43 IP_Address Jun 16 22:21:43 2020 Hostname \%\%10OSPF/5/OSPF_NBR_CHG: -MDC=1; OSPF 1 Neighbor IP_Address(Vlan-interface4094) changed from FULL to DOWN.	Jun 16 22:21:43 IP_Address Jun 16 22:21:43 2020 Hostname \%\%10OSPF/5/OSPF_NBR_CHG: -MDC=1; OSPF 1 Neighbor IP_Address(Vlan-interface4094) changed from FULL to DOWN.

Fig. 8. Clustered syslog for a specific fragment using our method and [8].

From the presented results in Fig. 7, different performance occur on two datasets which are reasonable. Firstly, The time-dense syslog in local testing networks could avoid more exhaustive search than time-sparse ones in real-world circumstances when DBSCAN is applied on time [2], which causes a significant drop in match efficiency in our second dataset. Moreover, the testing network aims to cover all correlations in the rule engine which involves many correlated syslog with low textual similarities. While in real scenarios, some interfaces might have correlated syslog from thousands of sub-interfaces with very similar textual structure which could significantly increase the hit rate in [8].

From the real-world observations, syslog in some periods of time show no successful matching for correlation rules since they are all isolated from each other. We extracted 2000 syslog (around 30-minute) from those and recorded the matching times for all three methods. 132,534, 132,534 and 3,408,046 times of rule matching were performed by our approach, [8] and [2] respectively. The results further tell that both ours and [8] showed good match efficiency while [2] acted more similar to exhaustive search.

Considering above unpredictable real-world circumstances and the high computational cost in finding proper parameter settings in DBSCAN algorithm in both two previous studies [2], [8], our method could still achieve acceptable match efficiency and high accuracy that guarantees our stable performance in practical usage under different real-world application scenarios without being influenced by parameter settings.

Furthermore, real-time capabilities were also evaluated where ours and [8] shared similar running time (around 3 seconds on average) for each 1-minute time window with almost stationary memory usage to ensure the stability of long-term operation. However, performing clustering by DBSCAN algorithm in [2] indicates its large time delay. It would be very hard to use DBSCAN to cluster syslog according to occurrence time when it is in a real-time scenario. Considering this, [2] is more suitable in mining correlation relationships with offline collected data instead of real-time network diagnosis.

V. Conclusion

In this paper, we proposed a training-free and parameter-free hierarchical tree-based syslog clustering method that could accurately cluster syslog according to both time and network constraints. The whole clustering process is completely automatic which largely reduce human efforts and increase the operation efficiency and could be applied to both offline and real-time scenarios. With the pre-processing, invalid and unnecessary syslog are initially filtered while only important ones are preserved for further clustering. The hierarchical method performs bottom-up clustering from leaves in the tree structure and merge time-correlated groups which could achieve high accuracy and efficiency.

To make the performance of our method more convincing, evaluations were conducted to assess the accuracy and efficiency aiming to make comparisons with other previous syslog clustering approaches. According to above results, our method showed outstanding performance in both hit rate (100%) and match efficiency (16%) in both local testing networks and real-world scenarios. In this way, our method could guarantee the precise grouping for further network diagnosis with enough known information. Moreover, our training-free and parameter-free method could provide real-time clustering and specific fault targeting by preserving hierarchical information which enables much more efficient network diagnosis.

However, our paper is currently limited to static parsing of syslog which requires regular update if the templates change. Considering this, dynamic parsing methods and incremental template learning would be employed to adapt our approach to more real-world scenarios.

Although above limitation is still being optimised, this scheme has already been applied in the Seer Analyzer product of H3C Company which is used for intelligent syslog management in root cause analysis and has achieved excellent performance.

Acknowledgment

We would like to thank colleagues from Intelligent Analyzer Development Department of H3C for providing real-world syslog data and AI Institute of H3C for providing computing resource to support this project.

REFERENCES

[1] D. Gürer and E. Lusher, "Method and system for fault diagnosis in a data network," Oct. 10 2006, uS Patent 7,120,819.

[2] B. Lu, N. Hua, X. Zheng, and W. Chen, "Application of network alarm association analysis processing based on artificial intelligence," *Designing Techniques of Posts and Telecommunications*, no. 12, pp. 1–6, 2018.

[3] S. Leonhardt and M. Ayoubi, "Methods of fault diagnosis," *Control engineering practice*, vol. 5, no. 5, pp. 683–692, 1997.

[4] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in *2017 IEEE International Conference on Web Services (ICWS)*. IEEE, 2017, pp. 33–40.

[5] S. Zhang, W. Meng, J. Bu, S. Yang, Y. Liu, D. Pei, J. Xu, Y. Chen, H. Dong, X. Qu *et al.*, "Syslog processing for switch failure diagnosis and prediction in datacenter networks," in *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*. IEEE, 2017, pp. 1–10.

[6] S. Zhang, Y. Liu, W. Meng, J. Bu, S. Yang, Y. Sun, D. Pei, J. Xu, Y. Zhang, L. Song *et al.*, "Efficient and robust syslog parsing for network devices in datacenter networks," *IEEE Access*, vol. 8, pp. 30 245–30 261, 2020.

[7] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 2016, pp. 102–111.

[8] N. Zhao, J. Chen, X. Peng, H. Wang, X. Wu, Y. Zhang, Z. Chen, X. Zheng, X. Nie, G. Wang *et al.*, "Understanding and handling alert storm for online service systems," in *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2020, pp. 162–171.

[9] L. Xu, J. Li, Z. Zhu, H. Chen, and L. Wang, "Research on pattern mining of alarm sequence based on network and time constraints," *Computer and Digital Engineering*, vol. 47, no. 9, pp. 2364–2368, 2019.

[10] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, 2009, pp. 117–132.

[11] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun *et al.*, "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs." in *IJCAI*, 2019, pp. 4739–4745.

[12] X. Yuan, "An improved apriori algorithm for mining association rules," in *AIP conference proceedings*, vol. 1820, no. 1. AIP Publishing LLC, 2017, p. 080005.

[13] Y. Gashaw and F. Liu, "Performance evaluation of frequent pattern mining algorithms using web log data for web usage mining," in *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. IEEE, 2017, pp. 1–5.

[14] M. Yin, W. Wang, Y. Liu, and D. Jiang, "An improvement of fp-growth association rule mining algorithm based on adjacency table," in *MATEC Web of Conferences*, vol. 189. EDP Sciences, 2018, p. 10012.

[15] T. Jia, Y. Li, and Z. Wu, "Survey of state-of-the-art log-based failure diagnosis," *Journal of Software*, vol. 31, no. 7, pp. 1997–2018, 2020.

[16] S. Ghanbari, A. B. Hashemi, and C. Amza, "Stage-aware anomaly detection through tracking log points," in *Proceedings of the 15th International Middleware Conference*, 2014, pp. 253–264.

[17] R. Vaarandi, "A breadth-first algorithm for mining frequent patterns from event logs," in *International Conference on Intelligence in Communication Systems*. Springer, 2004, pp. 293–308.

[18] T. Reidemeister, M. Jiang, and P. A. Ward, "Mining unstructured log files for recurrent fault diagnosis," in *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*. IEEE, 2011, pp. 377–384.

[19] S. Du and J. Cao, "Behavioral anomaly detection approach based on log monitoring," in *2015 International Conference on Behavioral, Economic and Socio-cultural Computing (BESC)*. IEEE, 2015, pp. 188–194.

[20] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[21] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[22] L. Han, A. L. Kashyap, T. Finin, J. Mayfield, and J. Weese, "Umbc_ebiquity-core: Semantic textual similarity systems," in *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, 2013, pp. 44–52.

[23] Z. Zheng, Y. Li, and Z. Lan, "Anomaly localization in large-scale clusters," in *2007 IEEE International Conference on Cluster Computing*. IEEE, 2007, pp. 322–330.

[24] D. Deng, G. Li, J. Feng, and W.-S. Li, "Top-k string similarity search with edit-distance constraints," in *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, 2013, pp. 925–936.

[25] S. Sanjappa and M. Ahmed, "Analysis of logs by using logstash," in *Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications*. Springer, 2017, pp. 579–585.

[26] X. Nie, Y. Zhao, K. Sui, D. Pei, Y. Chen, and X. Qu, "Mining causality graph for automatic web-based service diagnosis," in *2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2016, pp. 1–8.