Efficient and Robust Trace Anomaly Detection for Large-Scale Microservice Systems

Shenglin Zhang, Zhongjie Pan, Heng Liu, Pengxiang Jin, **Yongqian Sun***, Jiaju Wang, Xueying Jia, Hui Yang, Yongqiang Zou，Dan Pei

Yongqian Sun

ISSRE, October 2023

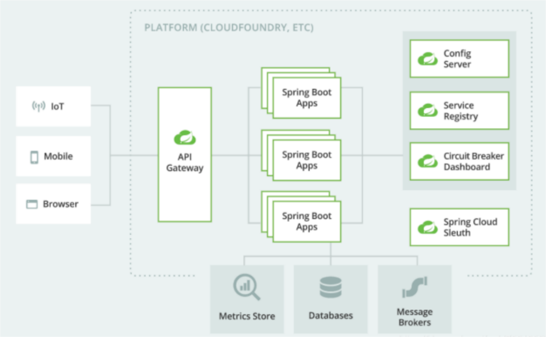
Outline

1. Background
2. Challenges
3. Framework
4. Evaluation
5. Conclusion
6. References

Outline

MicroService architecture

- Complex structure
- Large-scale instances
- Decentralization
- Loose coupling

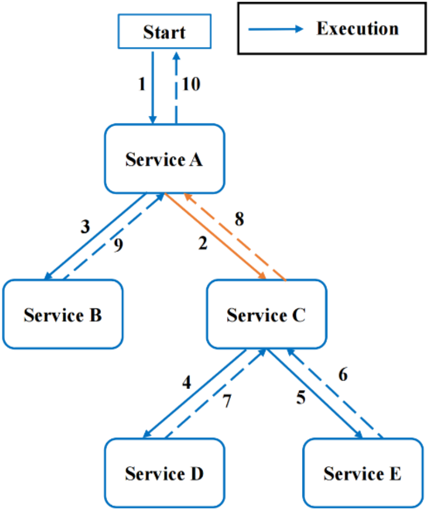


Impacts

- Company business operations gradually scale up.
- Enormous loss of system downtime.

COMPANY	ESTIMATED ANNUAL ECOMMERCE REVENUE	REVENUE PER HOUR	REVENUE PER SECOND	COST OF DOWNTIME THIS SESSION
Amazon.com	$115.88B	$13.22M	$3,671.98	$3.87M
WalMart.com	$21.44B	$2.45M	$679.52	$716,214.08
HomeDepot.com	$7.61B	$868,464	$241.24	$254,266.96
BestBuy.com	$6.13B	$698,832	$194.12	$204,602.48

What is a trace?



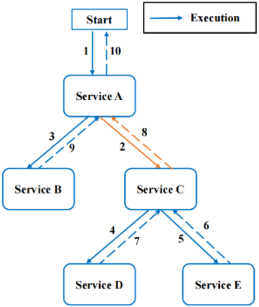
Execution 2 Recorded Information

traceID	3171c96e4f39c950
executionID	ux2sdgeppdmlzzl5q4h3l4x3ql4voz
parentID	61jtthdv7ydox6nlaieige3xg9d73jpf
start time	1642226961687440
end time	1642226961726055
function name	doCollectionPayFunc
end service name	Service C

Execution 8 Recorded Information

traceID	3171c96e4f39c950
executionID	sy1slu4inul0fi2smialwo8hiat4q5cv
parentID	6yam2kcju46lrs2nvysdu51ii3zrgtaj
start time	1642226961827245
end time	1642226961878004
function name	doCollectionReceiveFunc
end service name	Service A

Trace Features



Execution 2 Recorded Information

traceID	3171c96e4f39c950
executionID	ux2sdgeppdmlzzl5q4h3l4x3ql4voz
parentID	61jtthdv7ydox6nlaieige3xg9d73jpf
start time	1642226961687440
end time	1642226961726055
function name	doCollectionPayFunc
end service name	Service C

Execution 8 Recorded Information

traceID	3171c96e4f39c950
executionID	sy1slu4inul0fi2smialwo8hiat4q5cv
parentID	6yam2kcju46lrs2nvysdu51ii3zrgtaj
start time	1642226961827245
end time	1642226961878004
function name	doCollectionReceiveFunc
end service name	Service A

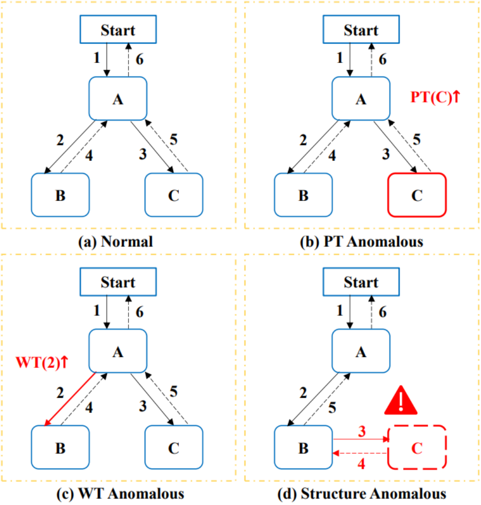
- Through calls between microservices, we can get invocation structure features.
- Through time features in execution, we can calculate the processing time(PT) at the services and the waiting time(WT) at the executions:

$$PT(E) = ST(6) - ET(5)$$
$$WT(5) = ET(5) - ST(5)$$

Trace Features

We identify three types of common anomalies:

- processing time (PT),
- waiting time (WT),
- structural anomaly



Existing trace-based anomaly detection methods and weakness

Name	Data structure	Weakness
$AEVB$[1]	Long-short-term-memory	Only foncus on time anomaly detection
$MultimodalTrace$[2]	Time series	Only focus on dependent and parallel task
$TraceAnomaly$[3]	Service trace vector	Fewer features used and long training time
$TraceCRL$[4]	Operation invocation graph	Coarse-grained anomaly detection and long training time

But neither of them can perform such fine-grained anomaly detection, or detection and root cause localization cannot be performed at the same time.

Outline

Challenge 1: Mixed normal and anomalous data

- Traces generated from company are mixed with anomalous traces.

Challenge 1: Mixed normal and anomalous data

- ? Traces generated from company are mixed with anomalous traces.

- **Ignore them and train straightly?**

  Learn the wrong pattern and decrease the recall.

THE IMPACT OF MIXED NORMAL AND ANOMALOUS DATA

Approach	F_1 trained on *cleaned* data	F_1 trained on *raw* data	Impact
MultimodalTrace [9]	0.809	0.337	0.472↓
AEVB [10]	0.831	0.328	0.503↓
TraceAnomaly [7]	0.828	0.385	0.443↓
TraceCRL [11]	0.860	0.427	0.433↓
Sage [12]	0.847	0.326	0.521↓

Challenge 1: Mixed normal and anomalous data

? Traces generated from company are mixed with anomalous traces.

- **Ignore them and train straightly?**

  Learn the wrong pattern and decrease the recall.

- **Split the anomalous traces from the normal traces?**

  The traces are unlabeled.

Challenge 2: Large data volume

? Traces are collected from a large-scale microservice system and recorded in large volume.

3 million traces each day in a middle enterprise,
Takes more than 192 hours (8 days) to train the model using one week data

Challenge 2: Large data volume

? Traces are collected from a large-scale microservice system and recorded in large volume.

- **Use partition to train model?**

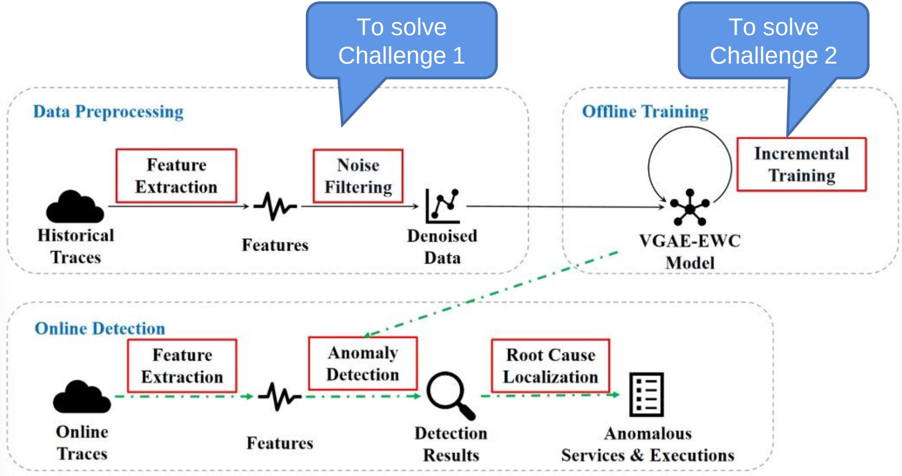
  Loss many patterns and decrease the precision.

- **Extract data using sampling strategies in statistical methods?**

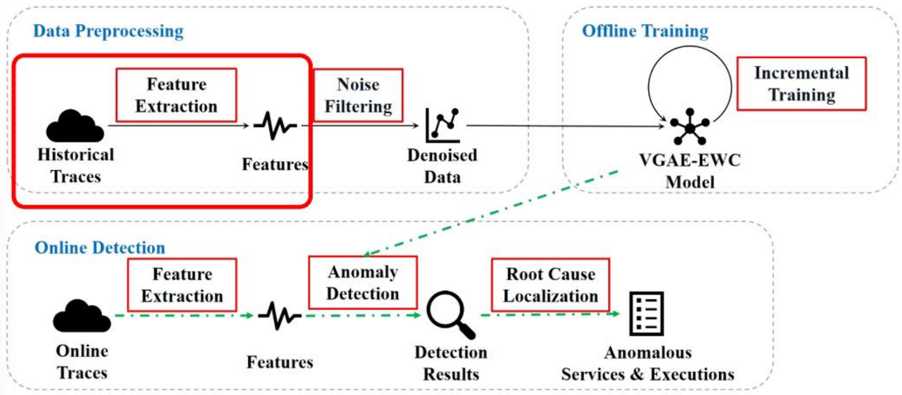
  The amount of different categories varies greatly, making it difficult to ensure the sampling is in accordance with the real data distribution.

Outline

The framework of TraceSieve

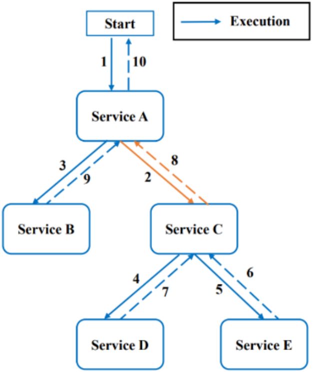


Feature extraction in the data preprocessing period



Three types of anomalies:

processing time (PT), waiting time (WT), structural anomalous



A trace

Trace Feature Matrix (TFM)

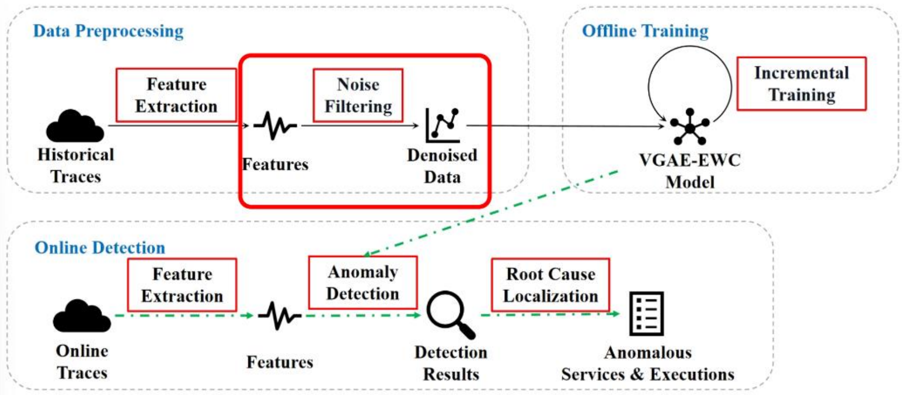
Waiting Time	Processing Time
WT(1)	0
WT(2)	$PT_1(A)$
WT(3)	$PT_2(A)$
WT(4)	$PT_1(C)$
WT(5)	$PT_2(C)$
WT(6)	PT(E)
WT(7)	PT(D)
WT(8)	$PT_3(C)$
WT(9)	PT(B)
WT(10)	$PT_3(A)$

Adjacency Matrix

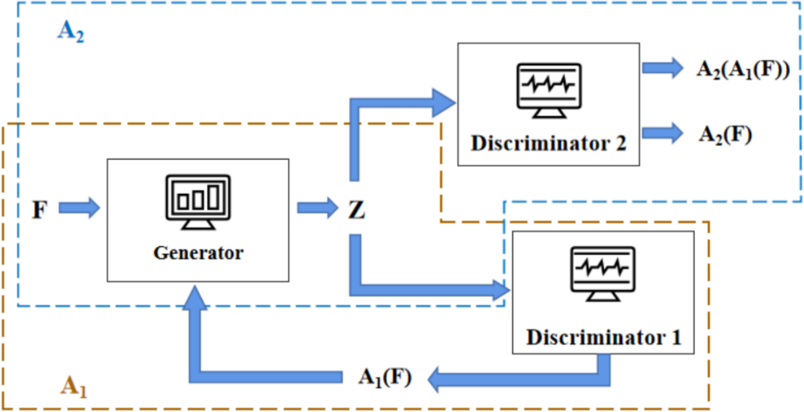
0	1	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	1	1
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0	0

The features of a trace

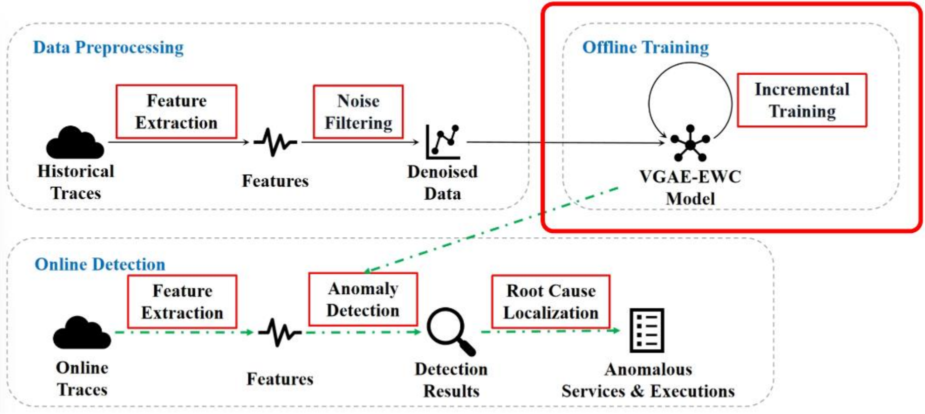
Noise filtering in the data preprocessing period



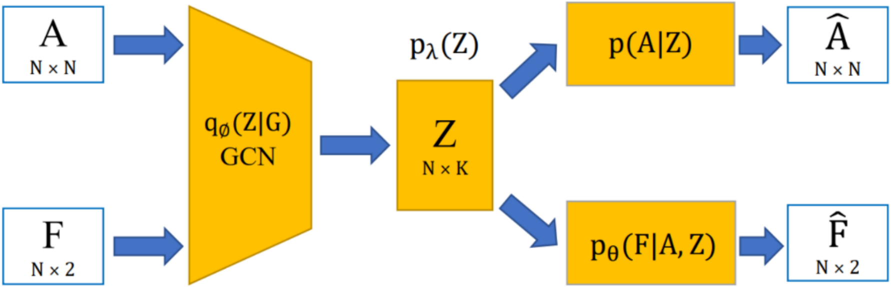
The detailed framework of noise filtering



VGAE-EWC in the offline training period



The detailed framework of VGAE-EWC

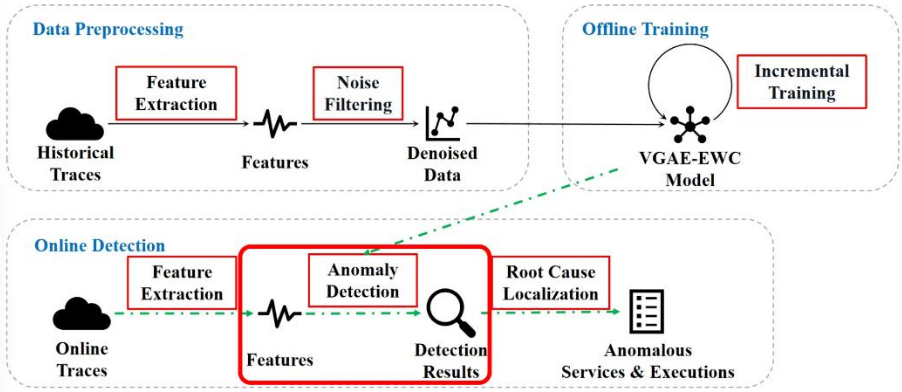


;

- Use the incremental training strategy called **Elastic Weight ConsoUidation** (EWC)[5], and the focus is on minimizing the loss function as follows:

$$L(\theta) = L_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2$$

Anomaly detection in the online detection preprocessing period



The details of anomaly detection

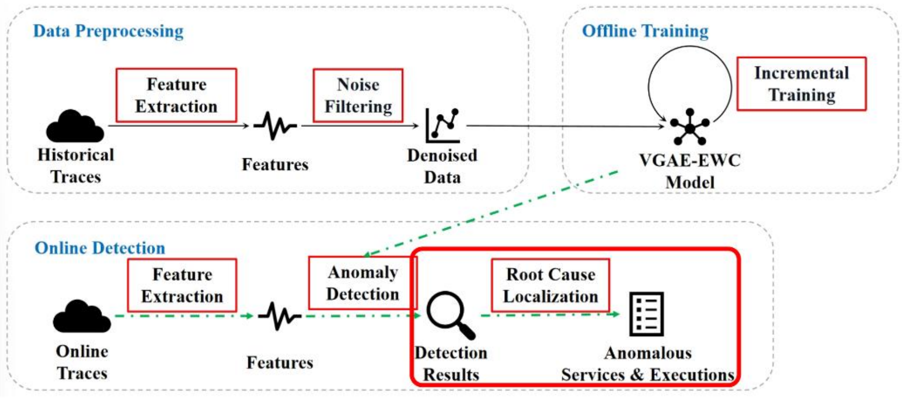
- TraceSieve trains a fine-tuned VGAE-EWC model for online detection of anomalies in new trace data.

- Use **negative log-likelihood (NLL)** as an anomaly score to each trace to discern whether it is anomalous:

$$
\begin{aligned}
NLL_G &= -\log p_{model}(G) \\
&= -\log \mathbb{E}q\phi(z|G,N)\left[\frac{p_\theta(G,N,z)}{q_\phi(z|G,N)}\right] \\
&\approx -\log\left[\frac{1}{L}\sum_{l=1}^{L}\frac{p_\theta(N,A,X,z^{(l)})}{q_\phi(z^{(l)}|G,N)}\right]
\end{aligned}
$$

- Use the **p-value** approach to distinguish anomalous score and set the **p-value** threshold at **0.001**.

Root cause localization in the data preprocessing period



Root cause localization in the data preprocessing period

- **The mission of root cause localization is to identify the root microservice that caused the system failure.**
- **Use the physical significance of the trace feature matrix to achieve root cause localization:**
  - Identify the trace feature matrix with the longest common invocation path with the homogeneous trace feature matrix of the anomalous trace.
  - Use the **z-score** normalization strategy to measure the abnormality of the values in the anomalous trace's trace feature matrix:

$$Anomaly\ Score(x_i) = \frac{x_i - \mu_x}{\sigma_x}$$

$$\mu_x = \frac{\sum_{i \in N} x_i}{N}$$

$$\sigma_x = \frac{\sum_{i \in N} (x_i - \mu_x)^2}{N}$$

Outline

Dataset

Dataset 1 Public dataset provided by CloudWise

Type	Records	Failures
training set	23520998	1191820
testing set	7010	3778

Dataset 2 An e-commerce company

Type	Records	Failures
training set	36705835	8442
testing set	7117	3155

Baseline

Trace Anomaly Detection

- CFG[6]
- CPD[7]
- $AVEB$[1]
- $MultimodalTrace$[2]
- $TraceAnomaly$[3]
- $TraceCRL$[4]

Root Cause Localization

- $MEPFL$[8]
- $TraceAnomaly$[3]
- $TraceRCA$[9]
- $MicroRank$[10]
- $Sage$[11]

The effects of different methods in trace anomaly detection on Dataset 1

Method	Precision	Recall	F_1-score	Training Time(h)
CFG[6]	0.652	0.749	0.697	90
CPD[7]	0.478	0.682	0.562	96
$MultimodalTrace$[2]	0.747	0.807	0.776	126.7
$AEVB$[1]	0.634	0.687	0.659	683.2
$TraceAnomaly$[3]	0.867	0.819	0.842	315
$TraceCRL$[4]	0.895	0.824	0.874	159.6
TraceSieve	**0.973**	**0.968**	**0.970**	**4.3**

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F_1 - score = 2 \times \frac{precision \cdot recall}{precision + recall}$$

The effects of different methods in trace anomaly detection on Dataset 2

Method	Precision	Recall	F_1-score	Training Time(h)
CFG[6]	0.610	0.722	0.661	46
CPD[7]	0.443	0.634	0.522	48
$MultimodalTrace$[2]	0.580	0.700	0.634	62.8
$AEVB$[1]	0.610	0.684	0.645	314.2
$TraceAnomaly$[3]	0.805	0.722	0.761	139.1
$TraceCRL$[4]	0.829	0.769	0.808	165.2
TraceSieve	**0.915**	**0.936**	**0.925**	**7.6**

The precision of different methods in root cause localization on Dataset 1

Method	Precision@1	Precision@2	Precision@3
$MEPFL$[8]	0.41	0.47	0.53
$TraceAnomaly$[3]	0.65	-	-
$TraceRCA$[9]	0.69	0.72	0.79
$MicroRank$[10]	0.76	0.83	0.88
$Sage$[11]	0.82	0.86	0.92
TraceSieve	**0.92**	**0.95**	**0.98**

The precision of different methods in root cause localization on Dataset 2

Method	Precision@1	Precision@2	Precision@3
MEPFL[8]	0.32	0.41	0.49
TraceAnomaly[3]	0.60	-	-
TraceRCA[9]	0.67	0.68	0.73
MicroRank[10]	0.72	0.83	0.85
Sage[11]	0.80	0.84	0.86
TraceSieve	**0.90**	**0.94**	**0.98**

Ablation Study

Dataset	Method	P	R	F_1	Time(h)
Dataset 1	w/o NFC	0.927	0.941	0.932	4.1
	w/o ITS	0.975	0.986	0.980	160.1
	TraceSieve	**0.973**	**0.968**	**0.970**	**4.3**
Dataset 2	w/o NFC	0.894	0.903	0.898	7.3
	w/o ITS	0.929	0.948	0.938	200.3
	TraceSieve	**0.915**	**0.936**	**0.925**	**7.6**

Outline

1. Background
2. Challenges
3. Framework
4. Evaluation
5. **Conclusion**
6. References

Conclusion

- We propose TraceSieve, a trace anomaly detection method to accurately detect anomalies for large-scale microservice system.
- Noise filtering component and incremental training strategy are combined to achieve accurate trace anomaly detection and less training time at the same time.
- Extensive evaluation experiments demonstrate that TraceSieve achieves more accuracy to other trace anomaly detection methods, and significantly outperforms existing methods in the speed of model training.

Outline

References

1. Sasho Nedelkoski and Jorge Cardoso and Odej Kao, Anomaly detec- tion and classification using distributed tracing and deep learning, in 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID 2019, Larnaca, Cyprus, May 14-17, 2019, 2019, pp.241-250.
2. S. Nedelkoski, J. Cardoso, and O. Kao, Anomaly detection from system tracing data using multimodal deep learning, in 12th IEEE International Conference on Cloud Computing, CLOUD 2019, Milan, Italy, July 8-13, 2019, 2019, pp.179-186.
3. P. Liu, H. Xu, Q. Ouyang, R. Jiao, Z. Chen, S. Zhang, J. Yang, L. Mo, J. Zeng, W. Xue, and D. Pei, Unsupervised detection of microservice trace anomalies through service-level deep bayesian networks, in 31st IEEE International Symposium on Software Reliability Engineering, ISSRE 2020, Coimbra, Portugal, October 12-15, 2020, 2020, pp.48-58.
4. C. Zhang, X. Peng, T. Zhou, C. Sha, Z. Yan, Y. Chen, and H. Yang, Tracecrl: contrastive representation learning for microservice trace analysis, in Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2022, pp.1221-1232.
5. J. Kirkpatrick, R. Pascanu, N. C. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, Overcoming catastrophic forgetting in neural networks, CoRR, vol. abs/1612.00796, 2016.
6. A. Nandi, A. Mandal, S. Atreja, G. B. Dasgupta, and S. Bhattacharya, Anomaly detection using program control flow graph mining from execution logs, in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, 2016, pp.215-224.
7. L. Bao, Q. Li, P. Lu, J. Lu, T. Ruan, and K. Zhang, Execution anomaly detection in large-scale systems through console log analysis, J. Syst. Softw., vol. 143, pp.172-186, 2018.
8. X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, D. Liu, Q. Xiang, and C. He, La- tent error prediction and fault localization for microservice applications by learning from system trace logs, in Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August 26-30, 2019, 2019, pp.683-694.
9. Z. Li, J. Chen, R. Jiao, N. Zhao, Z. Wang, S. Zhang, Y. Wu, L. Jiang, L. Yan, Z. Wang, Z. Chen, W. Zhang, X. Nie, K. Sui, and D. Pei, Practical root cause localization for microservice systems via trace analysis, in 29th IEEE/ACM International Symposium on Quality of Service, IWQOS 2021, Tokyo, Japan, June 25-28, 2021, 2021, pp.1-10.
10. G. Yu, P. Chen, H. Chen, Z. Guan, Z. Huang, L. Jing, T. Weng, X. Sun, and X. Li, Microrank: End-to-end latency issue localization with extended spectrum analysis in microservice environments, in WWW â21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021, 2021, pp.3087-3098.
11. Y. Gan, M. Liang, S. Dev, D. Lo, and C. Delimitrou, Sage: practical and scalable ml-driven performance debugging in microservices, in ASPLOS'21: 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Virtual Event, USA, April 19-23, 2021, 2021, pp.135-151.

南開大學 Nankai University

清華大學 Tsinghua University

Q&A

Thank you!

Name: **Yongqian Sun**

Email: sunyongqian@nankai.edu.cn

Date: **2023/10/16**