Guardian of the Resiliency: Detecting Erroneous Software Changes Before They Make Your Microservice System Less Fault-Resilient

Guanglei He[1, 4], Xiaohui Nie[2], Ruming Tang[3], Kun Wang[1, 4],
Zhaoyang Yu[1, 4], Xidao Wen[3], Kanglin Yin[3], Dan Pei[1, 4]

1 清華大學
2 中国科学院 计算机网络信息中心 Computer Network Information Center, Chinese Academy of Sciences
3 必示 BIZSEER
4 北京信息科学与技术国家研究中心 BEIJING NATIONAL RESEARCH CENTER FOR INFORMATION SCIENCE AND TECHNOLOGY

21 June 2024 // Guangzhou, IWQoS 2024

OUTLINE

BACKGROUND

FRAMEWORK

EVALUATION

SUMMARY

OUTLINE

BACKGROUND

FRAMEWORK

EVALUATION

SUMMARY

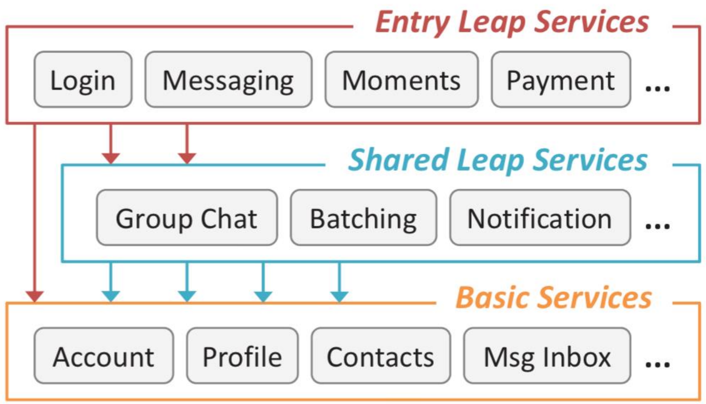
BACKGROUND

Microservice Architecture

Microservice architecture has become a staple in production.

Microservice Architecture
Arrange an application as a collection of loosely coupled, fine-grained services

Advantage: Scalability, Flexibility



Example: WeChat's microservice architecture*
3,000 services, over 20,000 nodes

*: Zhou, Hao, et al. "Overload control for scaling wechat microservices." *Proceedings of the ACM Symposium on Cloud Computing*. 2018.

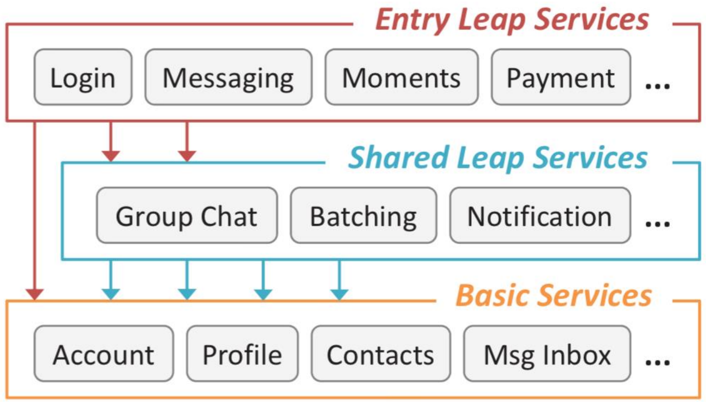
BACKGROUND

Microservice Architecture

Microservice architecture has become a staple in production.

Microservice Architecture
Arrange an application as a collection of loosely coupled, fine-grained services

Advantage: Scalability, Flexibility



Example: WeChat's microservice architecture*
3,000 services, over **20,000** nodes

Software changes in a microservice system

- **Frequent** but **error-prone**
- Google:
  70% of incidents were attributed to erroneous software changes

Software Change
Bug fixes, configuration adjustments, etc.

airbnb: **350+ per day**
Netflix: **4,000+ per day**
Google: **10,000+ per day**

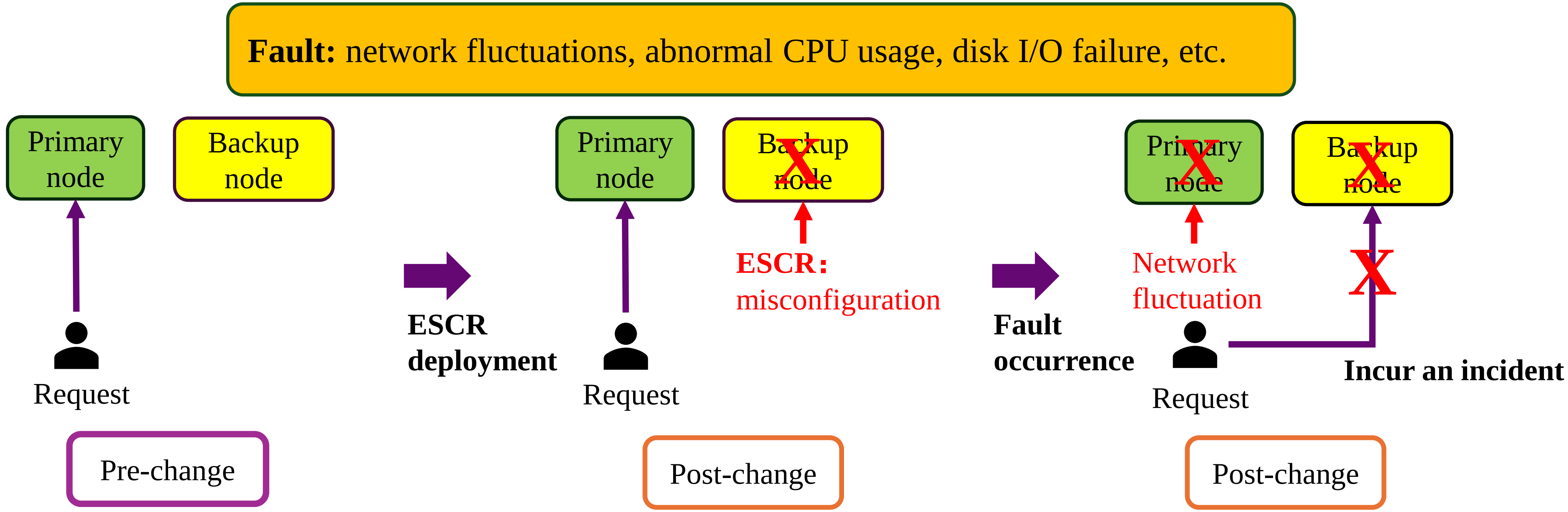
*: Zhou, Hao, et al. "Overload control for scaling wechat microservices." *Proceedings of the ACM Symposium on Cloud Computing*. 2018.

BACKGROUND

Erroneous Software Changes that Reduce fault Resilience

Some Erroneous Software Changes Reduce the fault Resilience (**ESCR**) of microservice systems.

- ESCRs incur incidents when faults occur in systems.

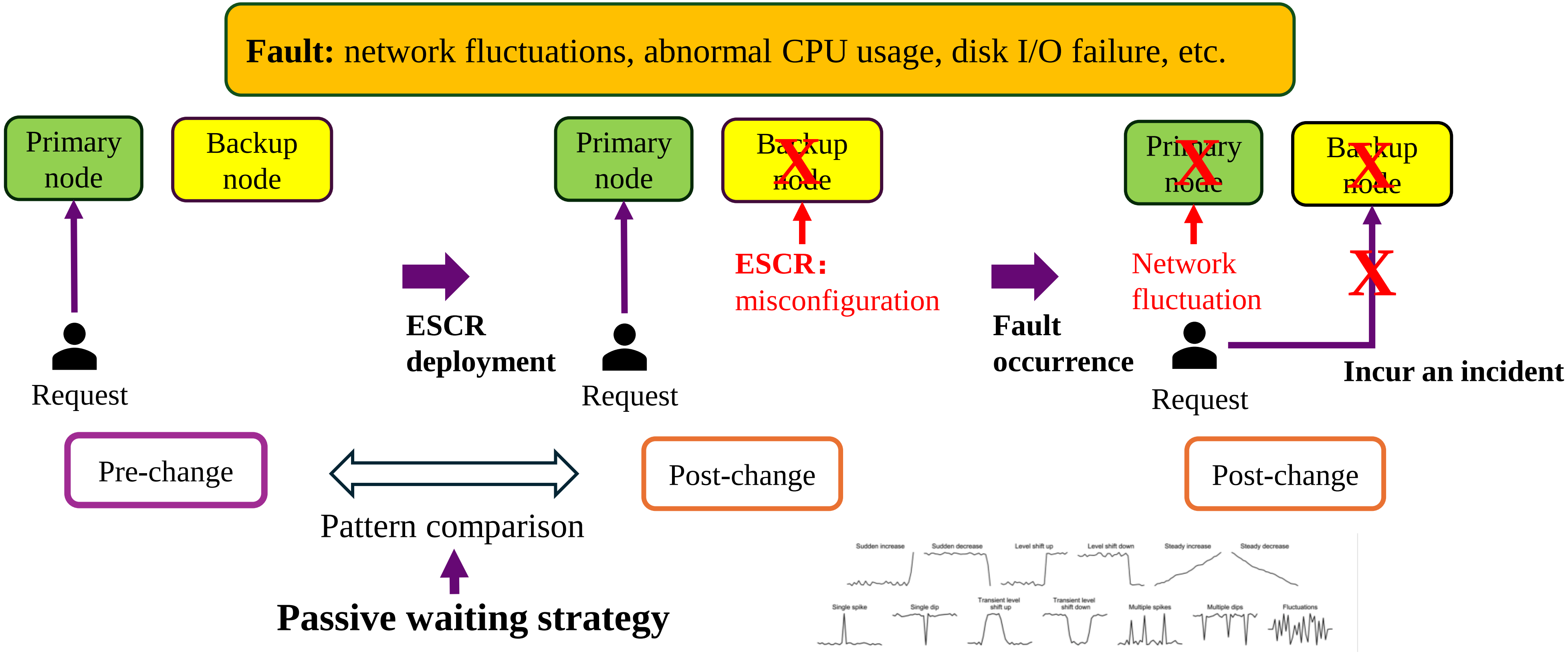


BACKGROUND

Erroneous **S**oftware **C**hanges that **R**educe fault **R**esilience

Some **E**rroneous **S**oftware **C**hanges **R**educe the fault **R**esilience (**ESCR**) of microservice systems.

- ESCRs incur incidents when faults occur in systems.
- Our empirical study reveals that **37.87% of the erroneous software changes qualified as ESCRs.**



Examples of patten variation

BACKGROUND

Challenge

Lack of training data

- **Insufficient** real-world abnormal data
- The high **labeling cost** of data generated through fault injection

Complex KPI patterns

- **Dozens of faults** should be injected to test the fault resilience
- Faults can **affect** the pattern of KPIs

Significant overhead

- **Millions of KPIs** to be checked in real-world microservice systems
- **Training** overhead **& Detection** overhead

BACKGROUND

Challenge

Lack of training data

- **Insufficient** real-world abnormal data
- The high

Complex

ResilienceGuardian!

- **Dozens of faults** should be injected to test the fault resilience
- Faults can **affect** the pattern of KPIs

Significant overhead

- **Millions of KPIs** to be checked in real-world microservice systems
- **Training** overhead **& Detection** overhead

OUTLINE

BACKGROUND

FRAMEWORK

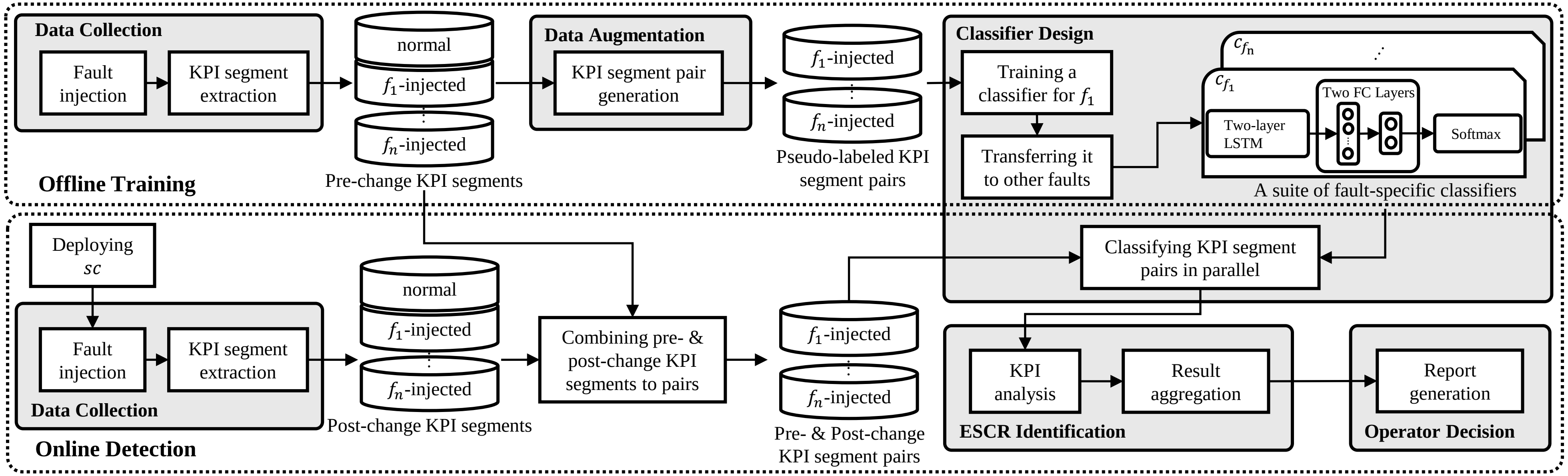
EVALUATION

SUMMARY

FRAMEWORK

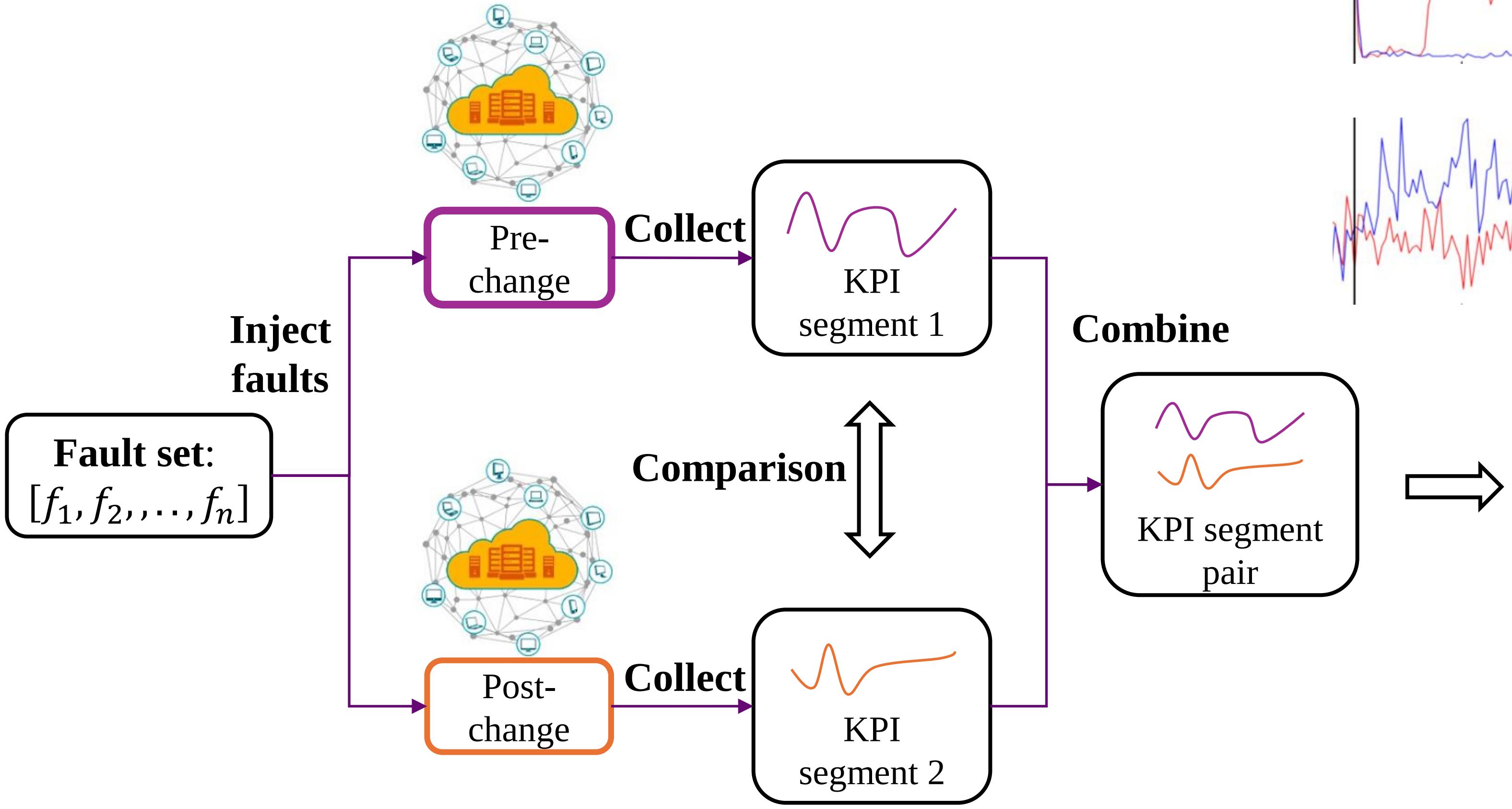
ResilienceGuardian

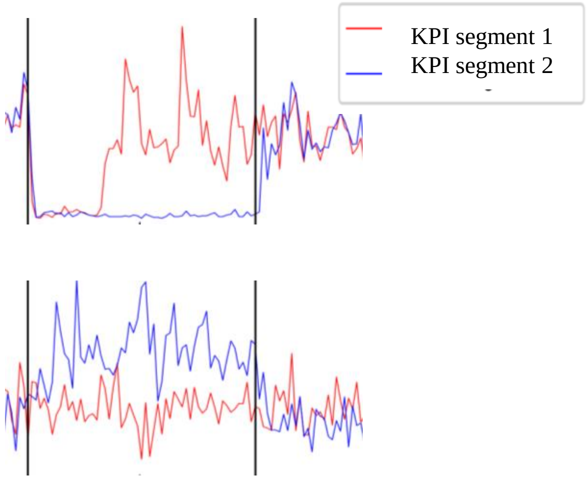
- Deploy the software change in the **staging environment**.
- Perform **fault injection** to test the resilience.
- Utilize **machine learning** models to process **KPI** data, aiming to assess the fault resilience.



FRAMEWORK

Data Collection





Classifications of KPI segment Pairs

- **Criteria:** Variations in KPI segment patterns

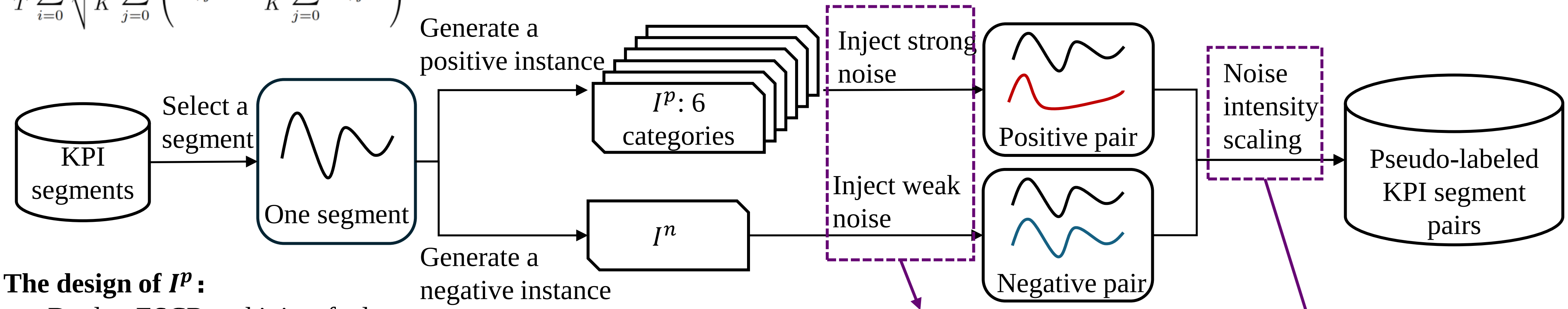
FRAMEWORK

Lack of training data

Data Augmentation

Noise Intensity: quantify the random noise in a KPI segment

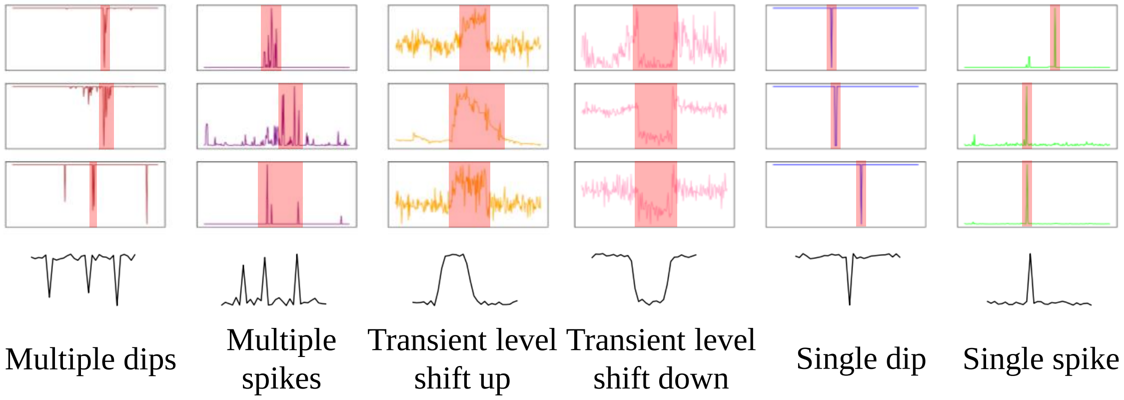
$$NI_X = \frac{1}{T}\sum_{i=0}^{T-1}\sqrt{\frac{1}{K}\sum_{j=0}^{K-1}\left(x_{i+j\times T} - \frac{1}{K}\sum_{j=0}^{K-1} x_{i+j\times T}\right)^2}$$



The design of I^p :

- Deploy ESCR and inject faults
- Collect KPI data
- Cluster KPI segment pairs

***k*-σ rule**
Strong noise: (3 x NI, 10 x NI] → Positive Label
Weak noise: (0 x NI, 3 x NI] → Negative Label



*NI*s might vary tens of times in microservice systems.

The amount of injected noise varies.

$$x_{i\text{-}scaled} = x_i \times \frac{NI_{default}}{NI_X}$$

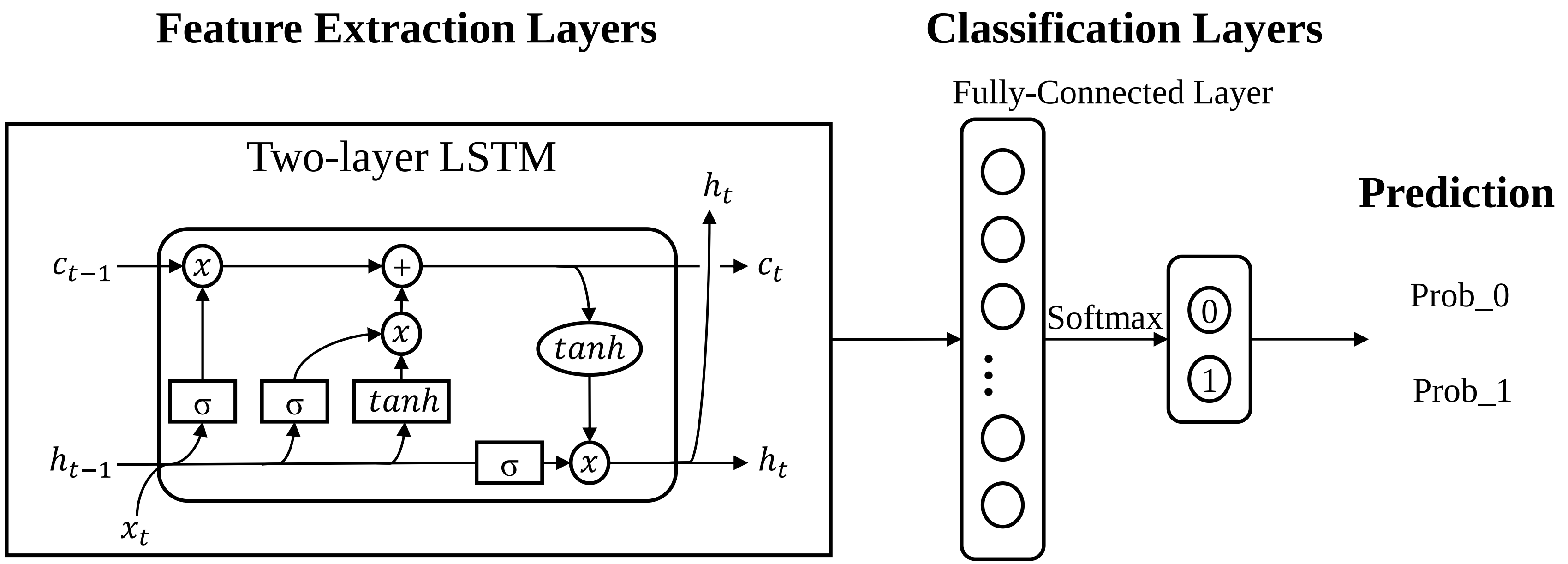
FRAMEWORK

Classifier Design: Model

- A **fault-specific** strategy: train individual classifiers for each fault.
- A **lightweight** deep-learning model

Complex KPI patterns

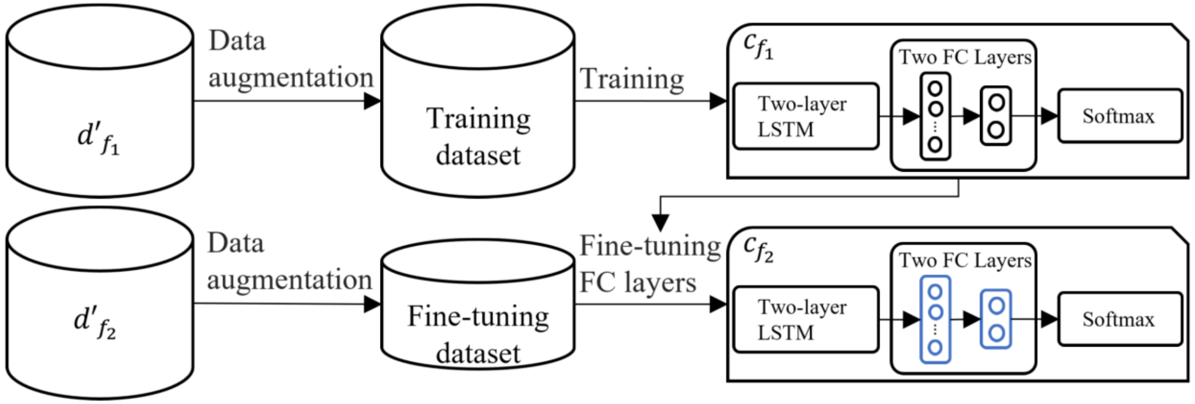
Significant overhead

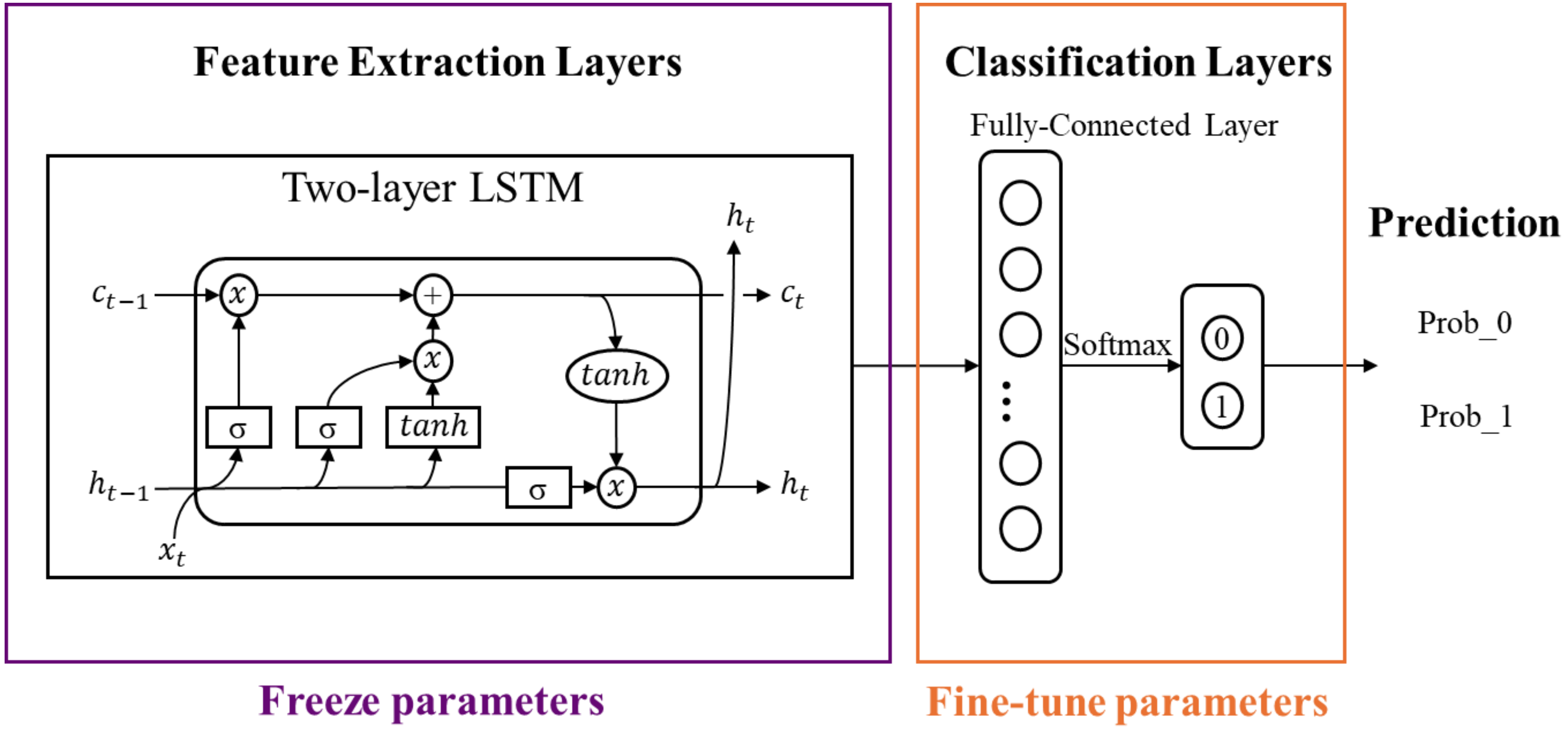


FRAMEWORK

Classifier Design: Transfer Learning

Significant training overhead



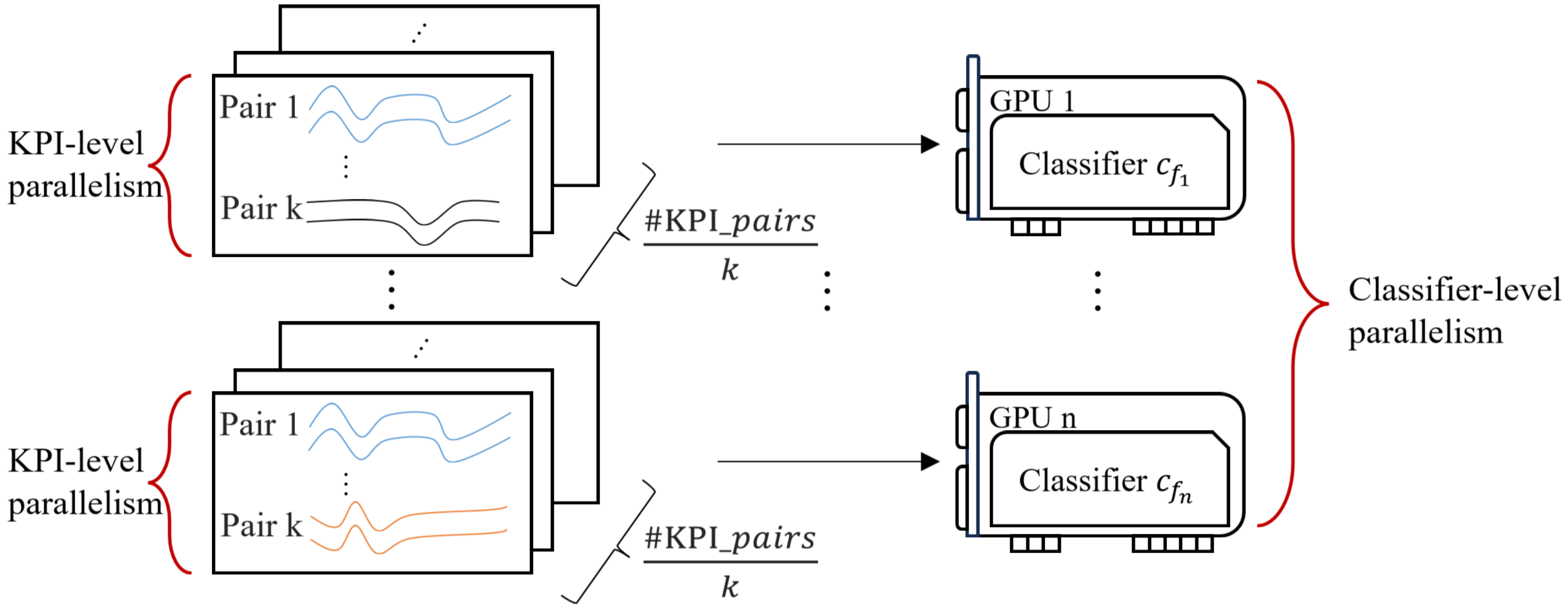


FRAMEWORK

Classifier Design: Parallelism Strategy

Strategy configuration: a 2-tuple (k, n)

Significant detection overhead



FRAMEWORK

ESCR Identification

Complex KPI patterns

KPI-level analysis

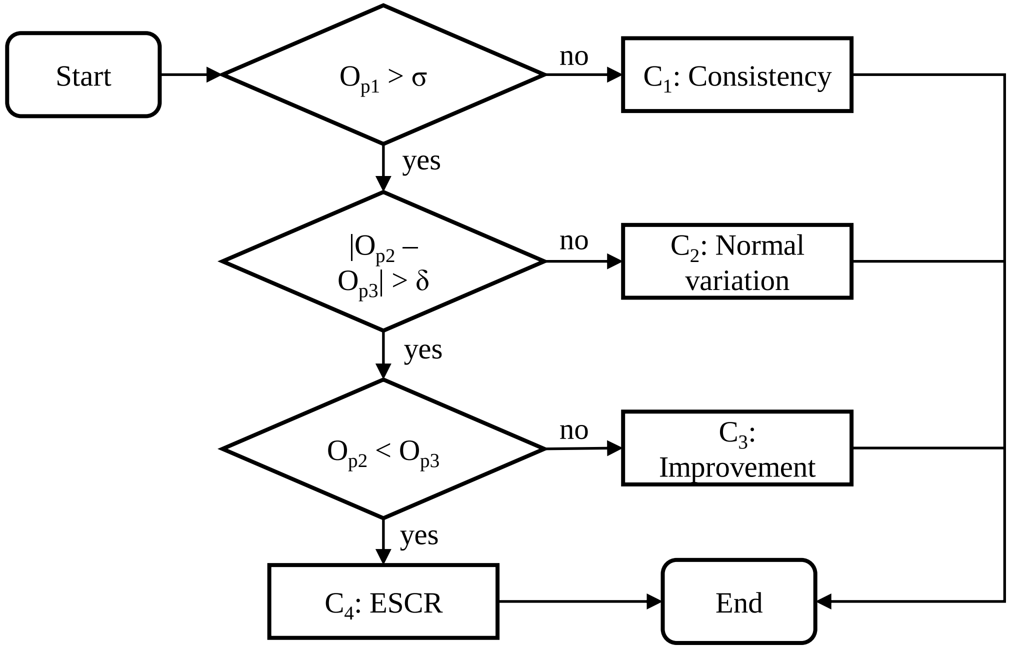
Classify each KPI's three pairs

Name	Segment 1	Segment 2
p_1	pre-change s'_f in d'_f	post-change s^{sc}_f in d^{sc}_f
p_2	pre-change s'_n in d'_n	pre-change s'_f in d'_f
p_3	post-change s^{sc}_n in d^{sc}_n	post-change s^{sc}_f in d^{sc}_f

Result aggregation:

Calculate a vulnerability score vs_f for each fault f

$$vs_f = \sum^{KPI \in C_4} \frac{|O_{p_2} - O_{p_3}|}{\sigma} O_{p_1}$$



FRAMEWORK

Operator Decision

- **Operator** should **confirm** these results.
- **Detection report**: records the KPI-level analysis and the aggregation result

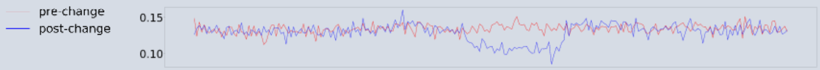
Software Change Ticket		Decision Panel
Service	HipsterShop-AdService	Accept Reject
Operation	Backup node modification of the ‘testbed-worker1’ server	
Submit Time	2023-09-30 17:00:00	
Recommendation	ESCR	

Detection Result

Rank 1 | **Vulnerability Score** 21.32 | **Fault** Network_Packet_Loss-50%_testbed-woker1 | **Start** 2023-09-30 21:00:00 | **Duration** 15 min

Index	Type	KPI	Category	Visualization
1	Business KPI	adservice-request_count	4	
2	Machine KPI	adservice-net_send_packet	4	

Rank 2 | **Vulnerability Score** 9.67 | **Fault** Abnormal_CPU_Usage-80%_testbed-woker1 | **Start** 2023-09-30 22:00:00 | **Duration** 15 min

Index	Type	KPI	Category	Visualization
1	Machine KPI	adservice-cpu_usage	4	
2	Business KPI	adservice-request_duration	4	

OUTLINE

BACKGROUND

FRAMEWORK

EVALUATION

SUMMARY

Evaluation

Datasets

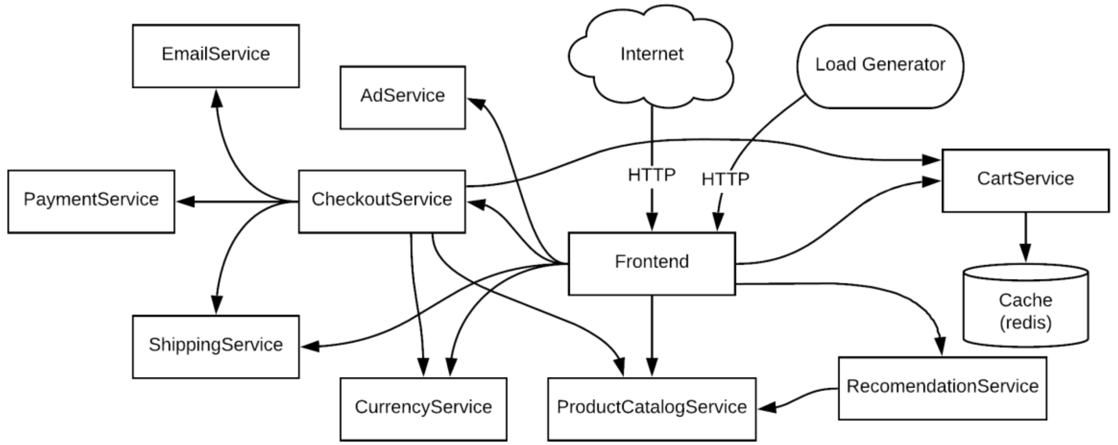
Dataset *A* & Dataset *B*

- Deploy ESCRs and perform fault injection
- **Dataset *A***: HipsterShop (**80** instances)
- **Dataset *B***: Train-Ticket (**120** instances)

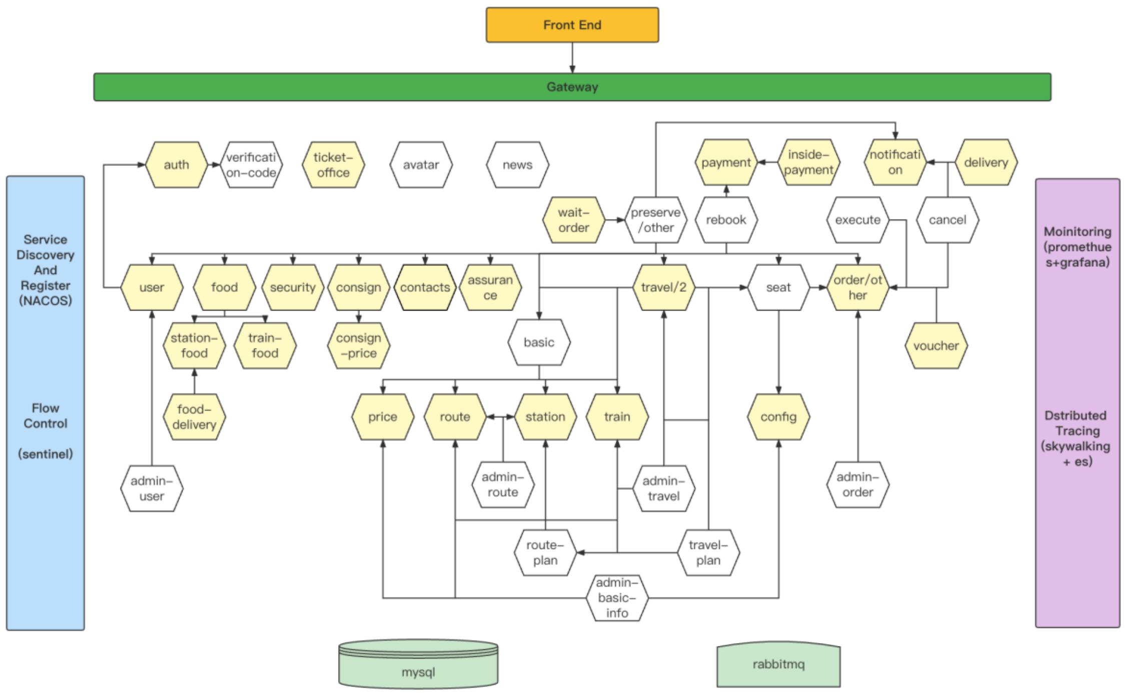
Category	Description
I	*E1* - Reducing CPU resources improperly *E2* - Reducing memory resources improperly
II	*E3* - Configuring insufficient CPU resources *E4* - Configuring insufficient memory resources
III	*E5* - Permitting expired requests to access invalid databases *E6* - Permitting expired requests to invoke death loops
IV	*E7* - Interrupting the forwarding of requests *E8* - Forwarding requests to invalid backup nodes

Dataset C

- The UEA Archive



HipsterShop Architecture*



Train-Ticket Architecture#

*: https://github.com/lightstep/hipster-shop
#: https://github.com/FudanSELab/train-ticket

Evaluation

ESCR Identification & Ablation Study

Dataset	Approach	P	R	F1	Training (min)	Detection (s)
$\mathcal{A}$	Gandalf	0.74	0.68	0.71	87.32	34.32
	SCWarn	0.64	0.59	0.61	23.91	9.35
	Kontrast	0.88	0.81	0.84	290.74	***0.10***
	Lumos	0.55	0.70	0.62	-	15.12
	Donut	0.78	0.54	0.64	327.86	17.24
	Telemanom	0.59	0.67	0.63	197.31	5.24
	ResilienceGuardian	***0.91***	***0.89***	***0.90***	***8.32***	0.12
$\mathcal{B}$	Gandalf	0.72	0.66	0.69	355.21	31.09
	SCWarn	0.69	0.65	0.67	77.36	38.18
	Kontrast	0.82	0.79	0.80	693.33	***0.11***
	Lumos	0.63	0.59	0.61	-	19.07
	Donut	0.72	0.67	0.69	1368.57	21.38
	Telemanom	0.55	0.58	0.56	814.89	5.32
	ResilienceGuardian	***0.87***	***0.92***	***0.89***	***33.86***	0.12

Dataset *A*: collected from **HipsterShop**
Dataset *B*: collected from **Train-Ticket**

ESCR Identification

- **F1 score: 0.91**
- **Training time:** a reduction of **56.23% - 97.53%**
- **Detection time: 0.12s**

Ablation Study

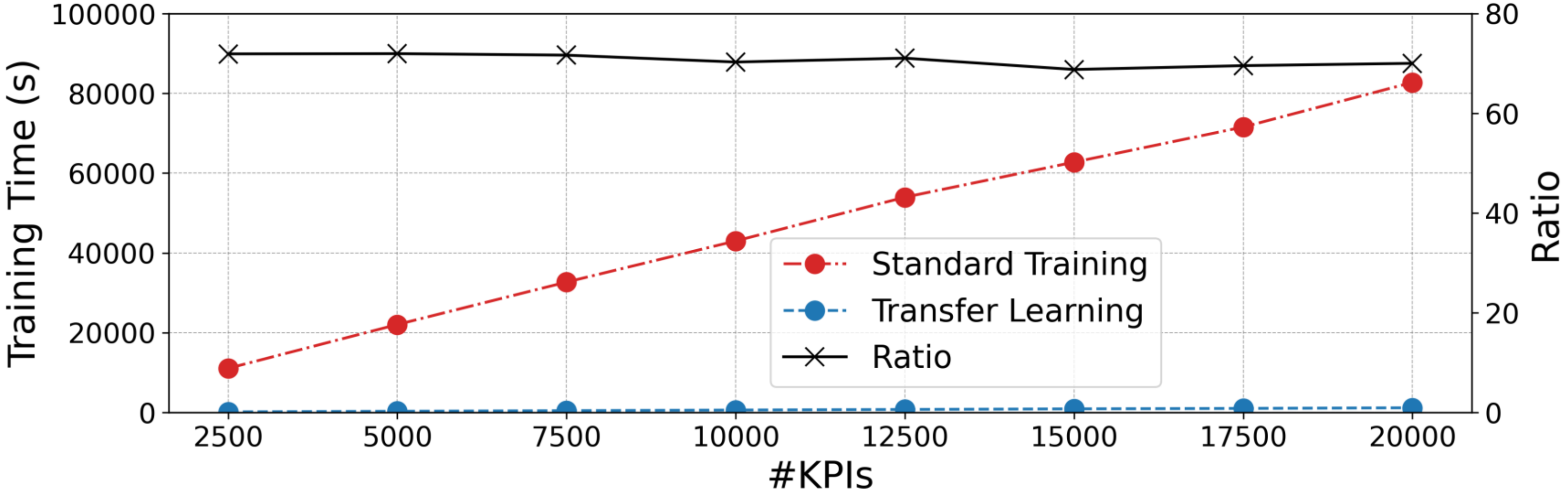
- Verify the contribution
  - Data augmentation
  - Transfer learning

Dataset	Approach	P	R	F1	Training (min)	Detection (s)
$\mathcal{A}$	***ResilienceGuardian***	0.91	0.89	0.90	8.32	0.12
	ResilienceGuardian$_{pre}$	0.79	0.83	0.81	9.42	0.12
	ResilienceGuardian$_{cate}$	0.77	0.87	0.82	20.85	0.13
	ResilienceGuardian$_{one}$	0.83	0.77	0.80	6.88	0.12
	ResilienceGuardian$_{all}$	0.94	0.91	0.92	68.79	0.12
$\mathcal{B}$	***ResilienceGuardian***	0.87	0.92	0.89	33.86	0.12
	ResilienceGuardian$_{pre}$	0.81	0.78	0.79	35.61	0.12
	ResilienceGuardian$_{cate}$	0.83	0.80	0.81	68.54	0.14
	ResilienceGuardian$_{one}$	0.76	0.87	0.81	31.09	0.13
	ResilienceGuardian$_{all}$	0.93	0.95	0.94	310.92	0.12

Evaluation

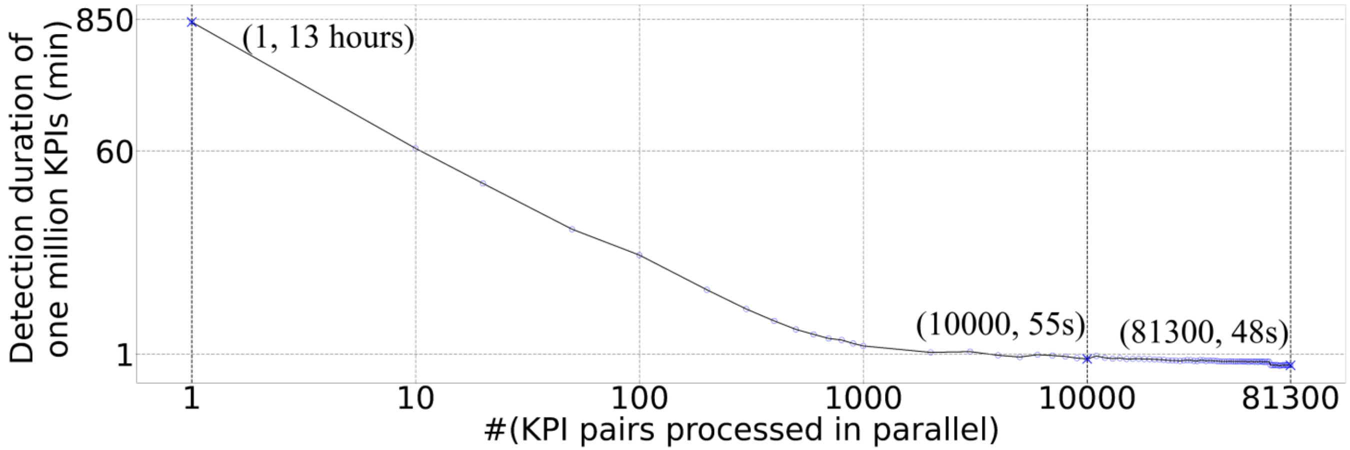
Scalability

Scale the dataset by duplicating KPI segment pairs.



Transfer Learning

- Ratio: **70**



Parallel Detection

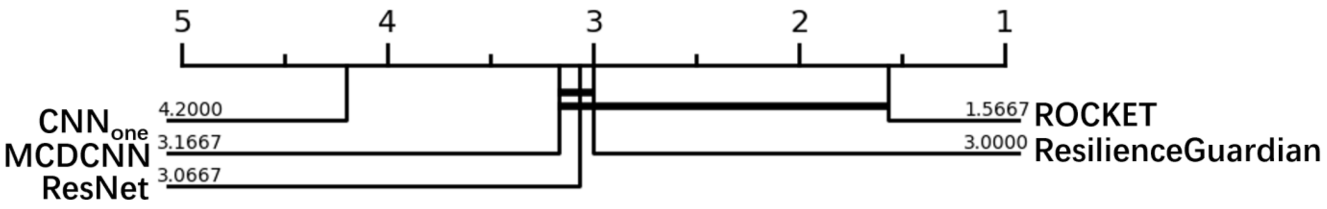
- Serial: **13 hours**
- Parallelism: **48s**

Evaluation

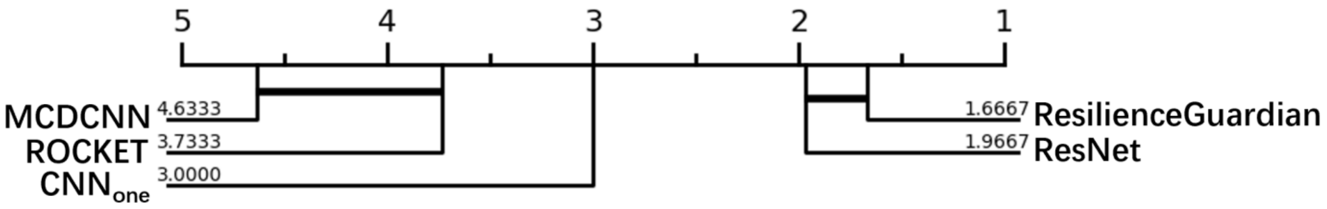
Classification

Critical Difference Diagram

- **Coordinate**: the mean rank of the model for all datasets



F1-Score



Running time

Dataset *C*: The UEA archive

Effectiveness

- **F1-Score: Similar** to ROCKET

Efficiency

- **Running time: Outperform** ROCKET

Conclusion

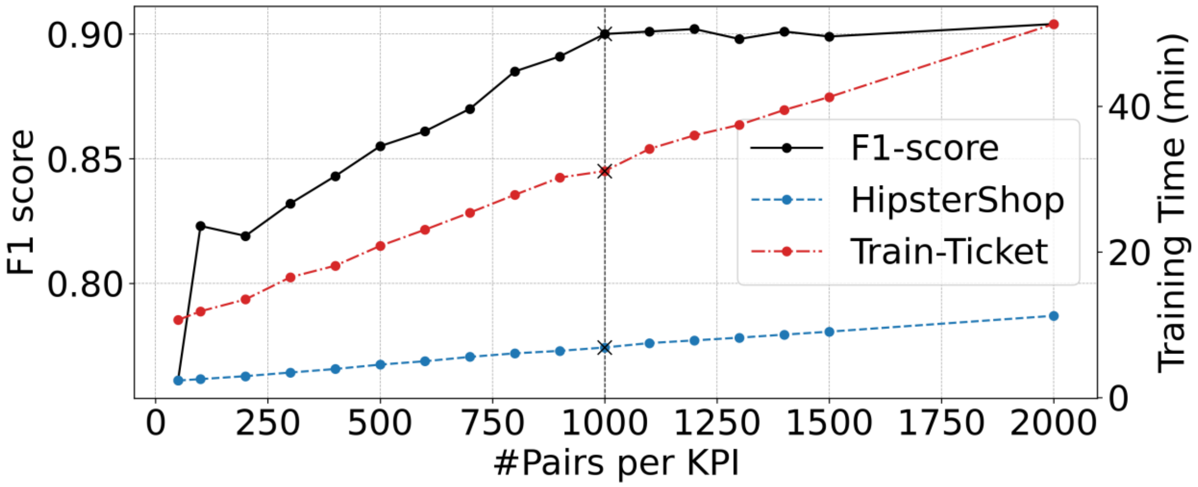
- A **satisfying balance** between effectiveness and efficiency

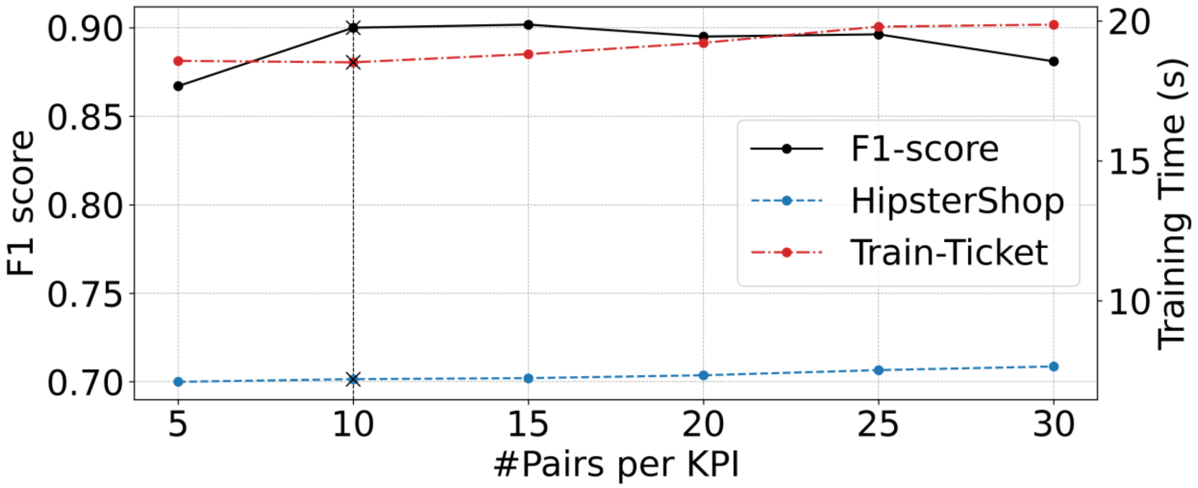
Evaluation

Hyperparameter Configuration

α: #Pairs per KPI for Training
- **α == 1000**

β: #Pairs per KPI for Transfer Learning
- **β == 10**





OUTLINE

BACKGROUND

FRAMEWORK

EVALUATION

SUMMARY

Summary

- To the best of our knowledge, this paper is **the first attempt** to address **Erroneous Software Changes that Reduce fault Resilience (**ESCR) identification.

- Our framework, ResilienceGuardian, enables **the early detection** of ESCRs before they impact the **fault resilience** of microservice systems in production.

- ResilienceGuardian is systematically evaluated on **two well-known microservice systems** , achieving an average **F1-score** of **0.90** in identifying ESCRs.

Thank you!
Q&A

hgl21@mails.tsinghua.edu.cn

IWQoS 2024