ARISE
Automated Reliable Intelligent Software Engineering

香港中文大學
The Chinese University of Hong Kong

HUAWEI

Huawei Cloud

MicroRes: Versatile Resilience Profiling in Microservices via Degradation Dissemination Indexing

T. Yang*, C. Lee*, J. Shen*, Y. Su‡, C. Feng†, Y. Yang†, and M. R. Lyu*

*The Chinese University of Hong Kong, †Huawei Cloud Computing Technology Co., Ltd, ‡Sun Yat-sen University

2024/9/21

Online Cloud Services' Reliability Is Crucial

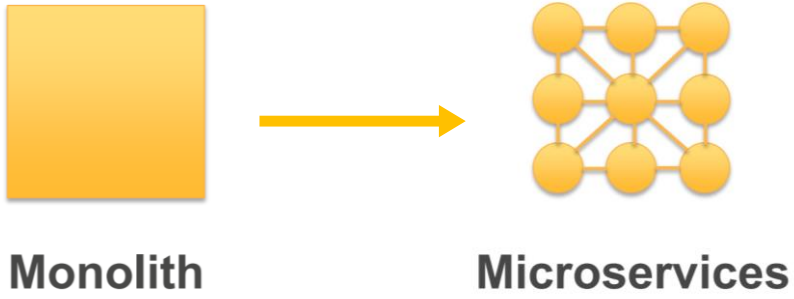
Twitter Back After Two-Hour Outage Affected Tweets

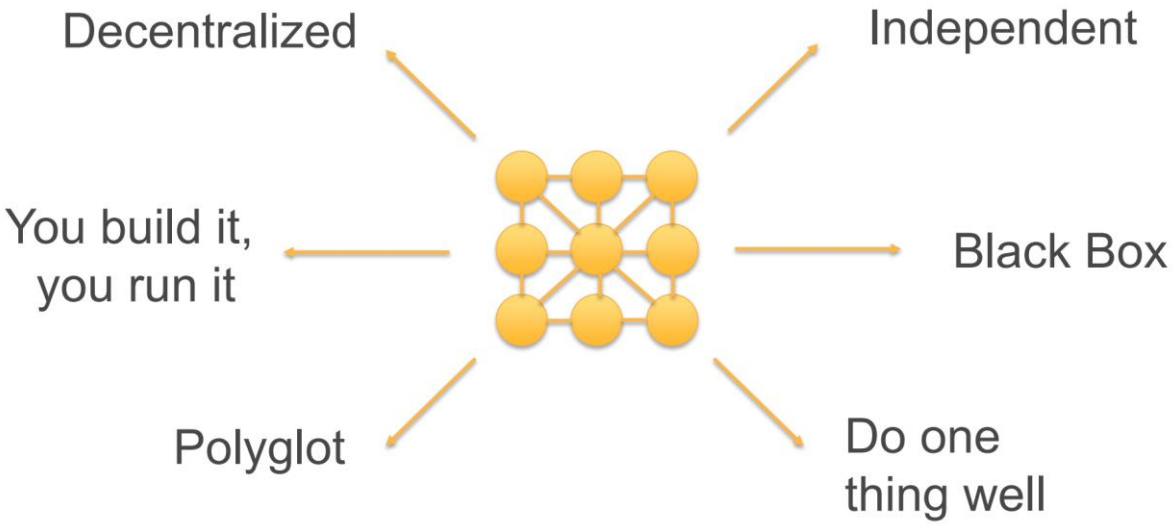
Facebook Lost About $65 Million During Hours-Long Outage

YouTube App Down on iOS Devices

Amazon's One Hour of Downtime on Prime Day May Have Cost It up to $100 Million in Lost Sales

Background: The Microservice Architecture





docker

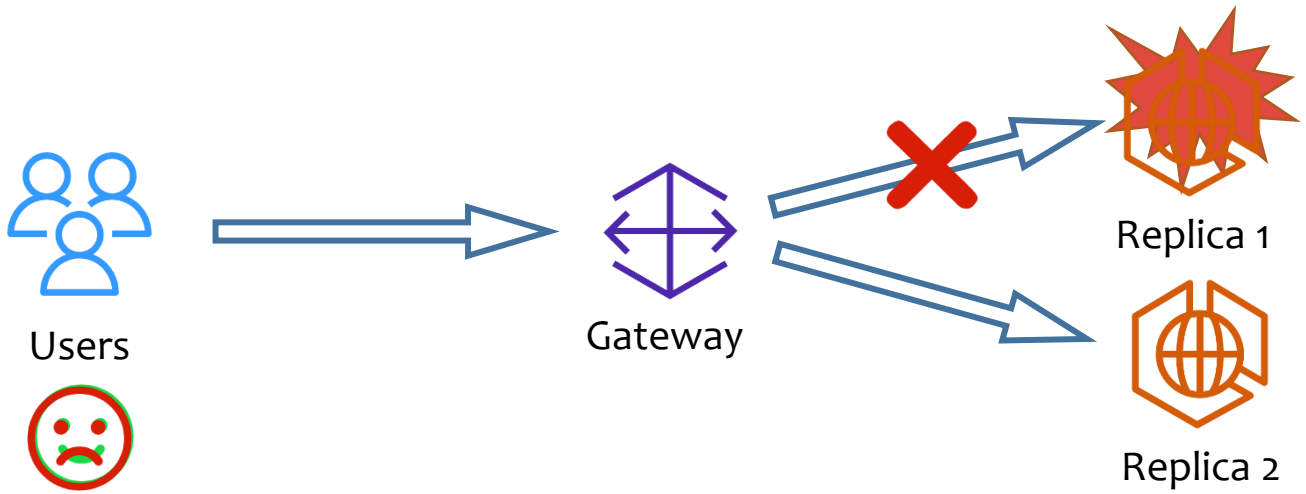
kubernetes

Microservices architecture is an approach in which a single application is composed of many **loosely coupled** and **independently deployable** small programs.

Microservices on AWS, AWS Summit Berlin 2016, Apr 12, 2016
What are Microservices? | IBM

Background: Resilience of Online Services

Resilience: the ability to maintain performance at an acceptable level and recover the service back to normal under service failures.



Test software resiliency - IBM Garage Practices

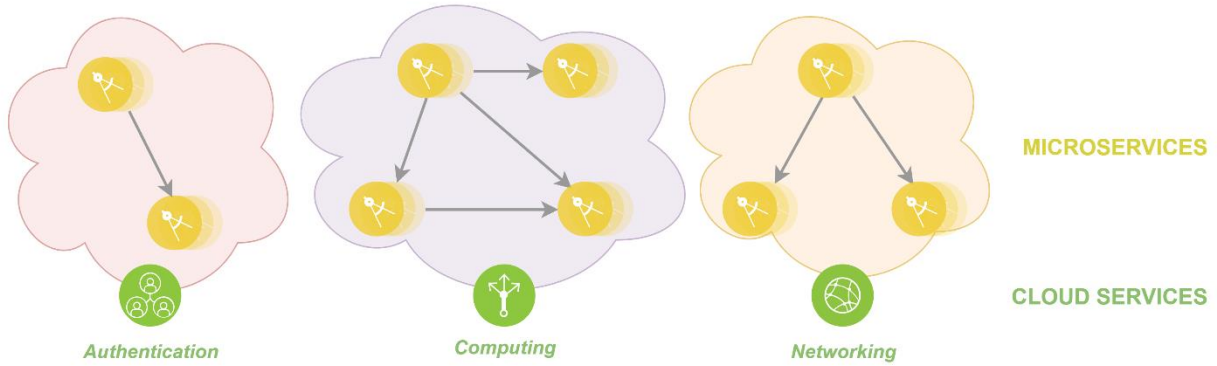
Background: Monitoring Metrics

- Monitoring Metrics
  - Observes real-time statuses of microservice systems.
  - Timestamped data with fixed intervals.

- Terminologies
  - System performance metrics.
    - E.g., CPU usage, memory usage, NIC send/receive rate.
  - User-aware metrics.
    - E.g., Request latency, request error rate, and throughput.

Online Service Systems Shift to Microservices

- Microservices collectively comprise multiple cloud services.
  - **Online services**: provide high-level APIs.
  - **Microservices**: collectively handle the external request via multiple chained invocations.



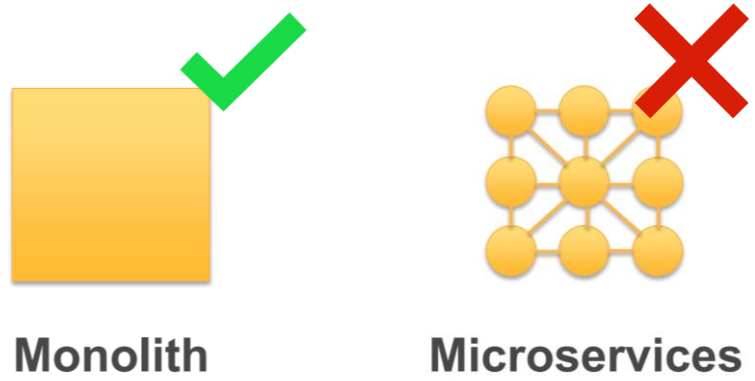
- Minor anomalies may magnify impact and escalate into system outages!

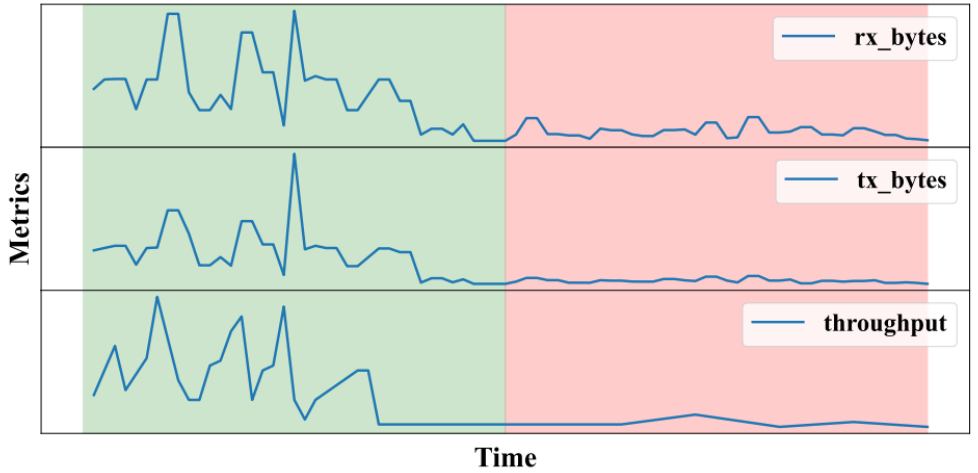
Loosely-coupled nature makes failure diagnosis difficult.

Current Practice for Resilience Testing

Failure type	Network jam
Metrics to monitor	Rx_bytes, tx_bytes, throughput
Passing criteria	Request throughput recover within 5 minutes

An example rule set





Issues of Current Practice: Labor-intensity

- Labor-intensity Issue
  - Microservices are highly decoupled with a large number of components.
  - Fast-evolving nature of microservices requires frequent updates of failure rule sets.

Manual identification of failure rule sets is too much labor-intensive.

Issues of Current Practice: Flexibility

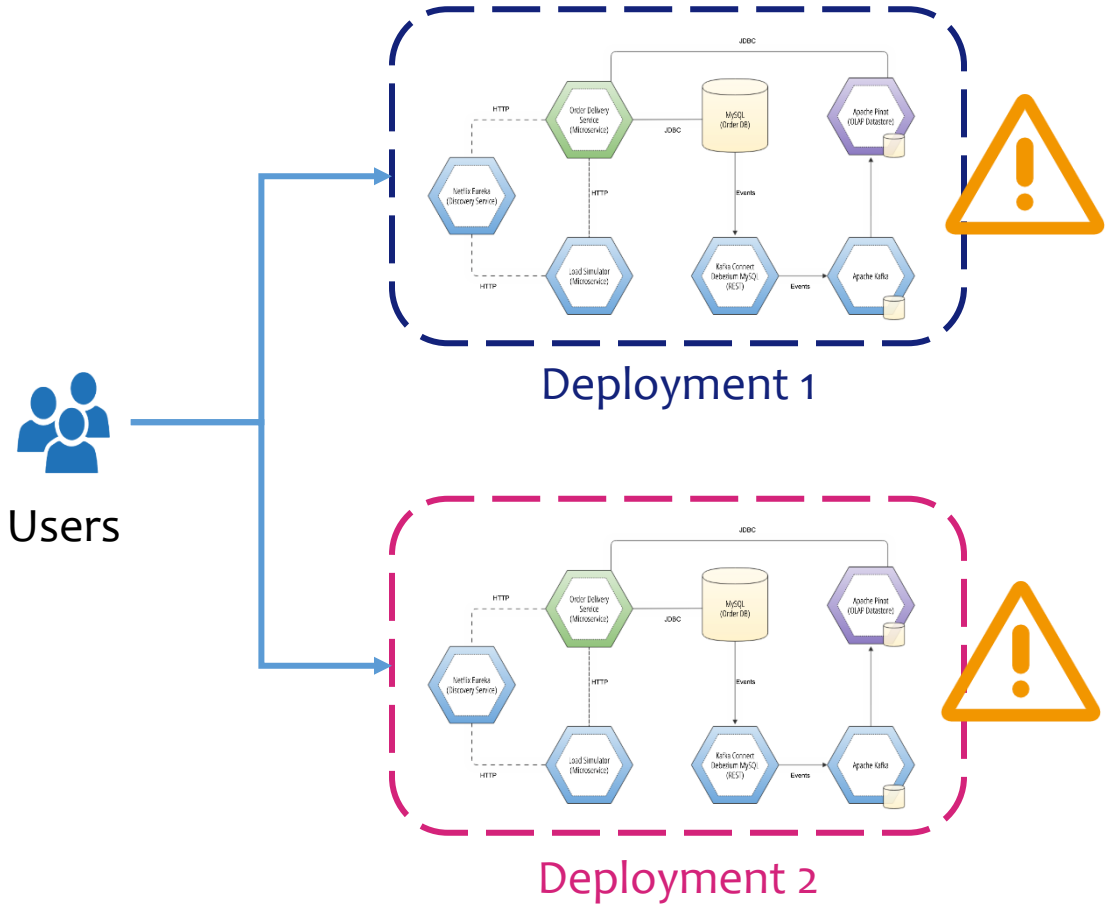
- Flexibility Issue
  - Microservices are specialized and may fail in different ways.
  - Fixed resilience test rules with binary PASS/FAIL results may not adequately capture the subtle differences in service resilience.

Defining fixed failure rule sets for evaluating resilience is not flexible.

Characteristics of Resilient Microservices

- Inject failures into two deployments of the same microservice benchmark system.
  - One **with common resilience measures**
  - One **without common resilience measures**

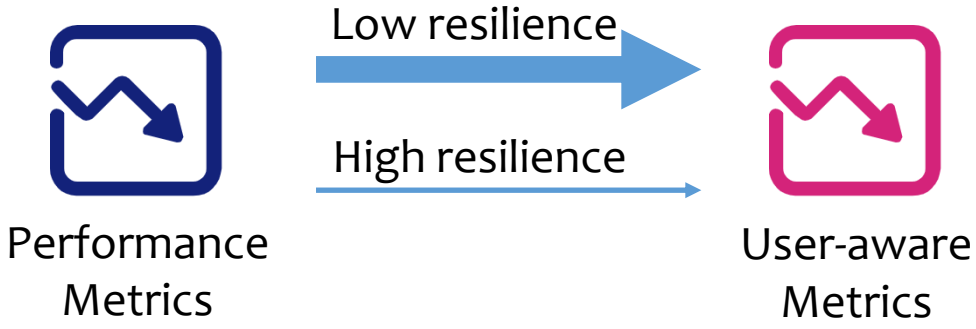
- Compare the manifestation of failures on the two deployments.



Characteristics of Resilient Microservices

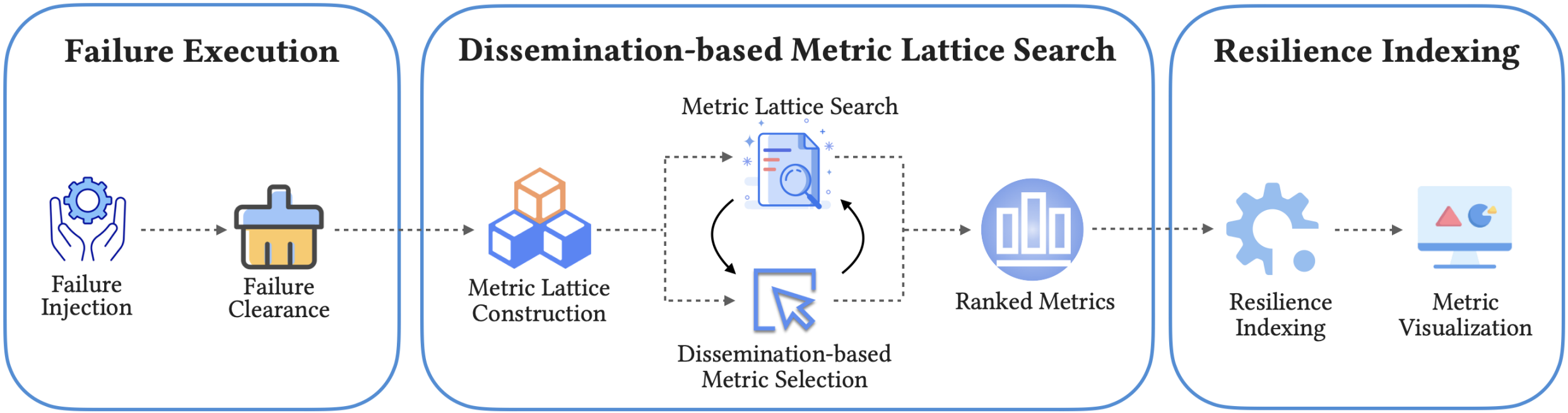
- Service degradation manifests the impact of the injected failures.
  - Measured by the performance difference between the normal period and the fault-injection period.

- **Insight**
  - The less degradation propagation from **system performance metrics** to **user-aware metrics**, the higher the resilience.



Failure	Degradation w/o resilience mechanisms	Degradation w/ resilience mechanisms
Container CPU overload	High container CPU usage, slow response speed	Decreased but acceptable response speed
Container TCP disconnection	Connection error within container	Return to normal response speed shortly
Container instance killed	Instance offline, unresponsive microservice endpoint	Response normally after some time
(More in the paper) ……		

MicroRes: A Versatile Resilience Profiling Framework

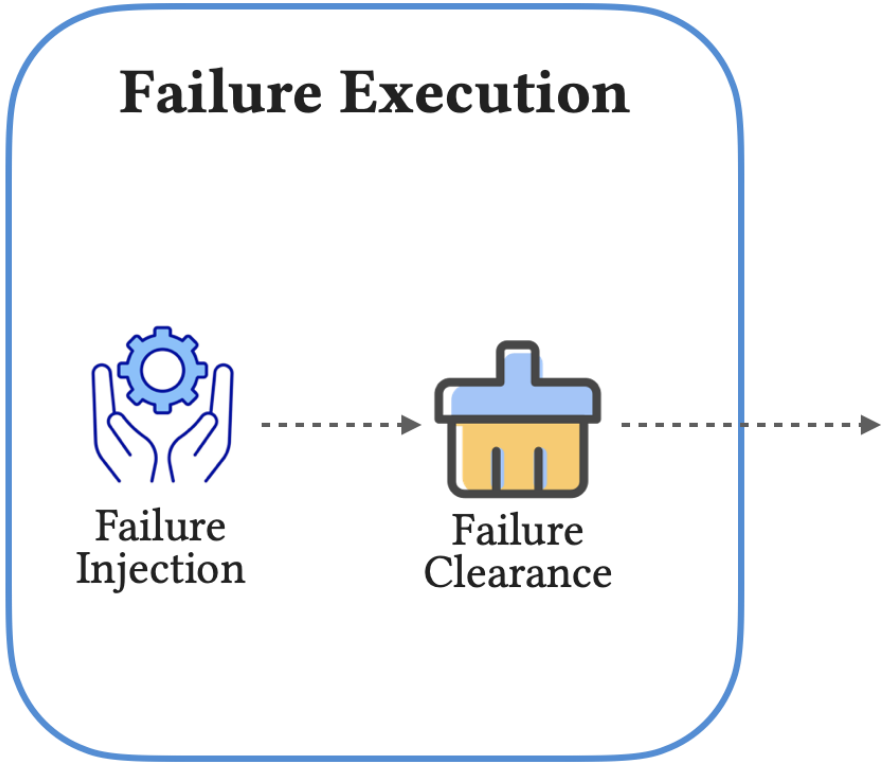


Collect monitoring metrics.

Rank the metrics by degradation.

Index the resilience.

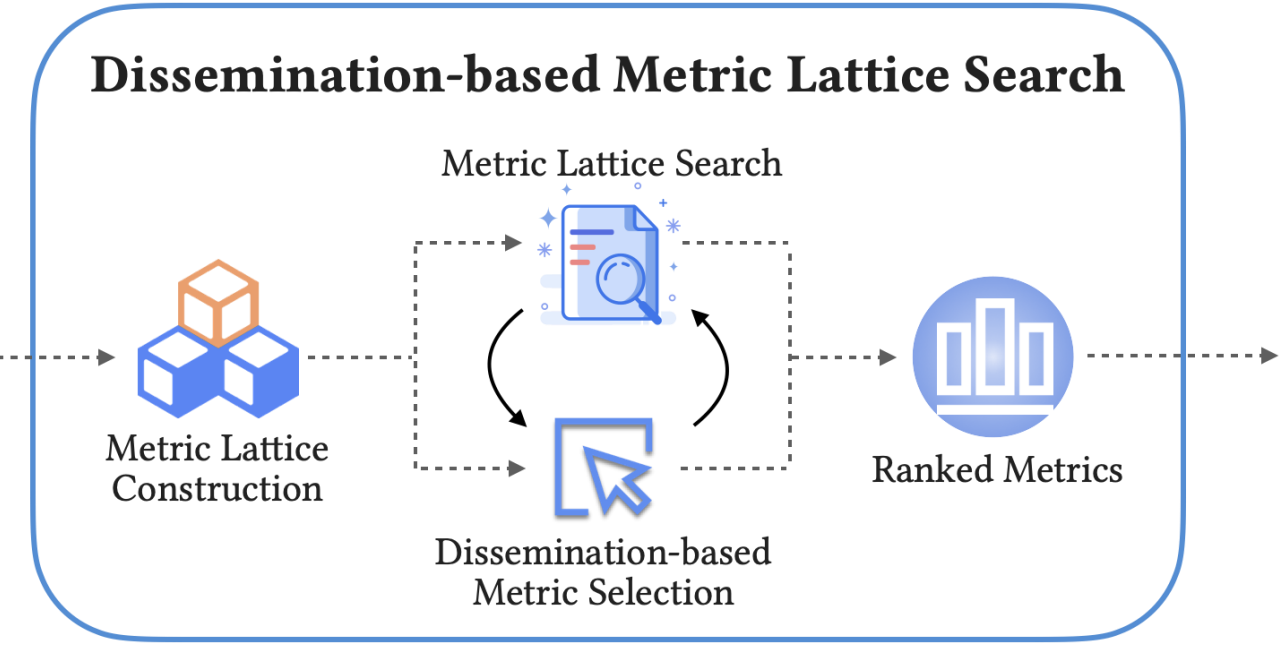
MicroRes: Failure Execution



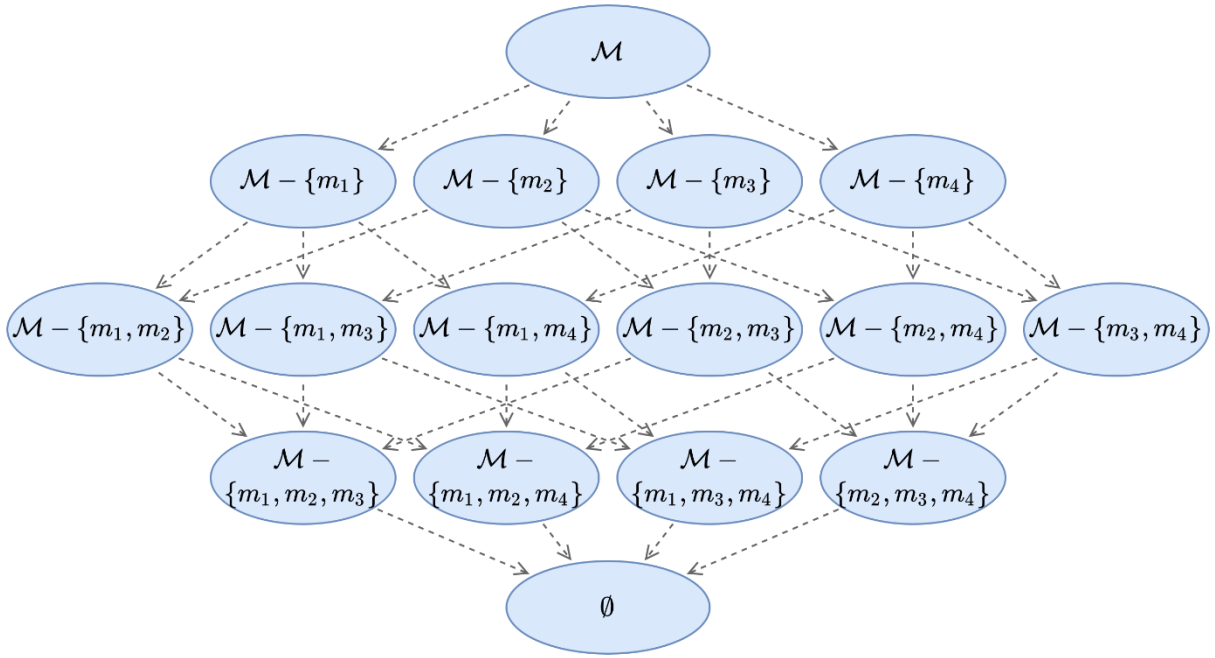
- Two phases for each type of failure.
  - Failure injection & Failure clearance.
- Data collected
  - Two types of metrics
    - User-aware metrics $\boldsymbol{U}$
    - System performance metrics $\boldsymbol{P}$
- Denote all metrics as $\boldsymbol{M}$

$$\boldsymbol{M} = \boldsymbol{U} \cup \boldsymbol{P} = \{m_1, m_2, \dots, m_M\}$$
$$\exists i, m_i \in \boldsymbol{U} \vee m_i \in \boldsymbol{P}$$

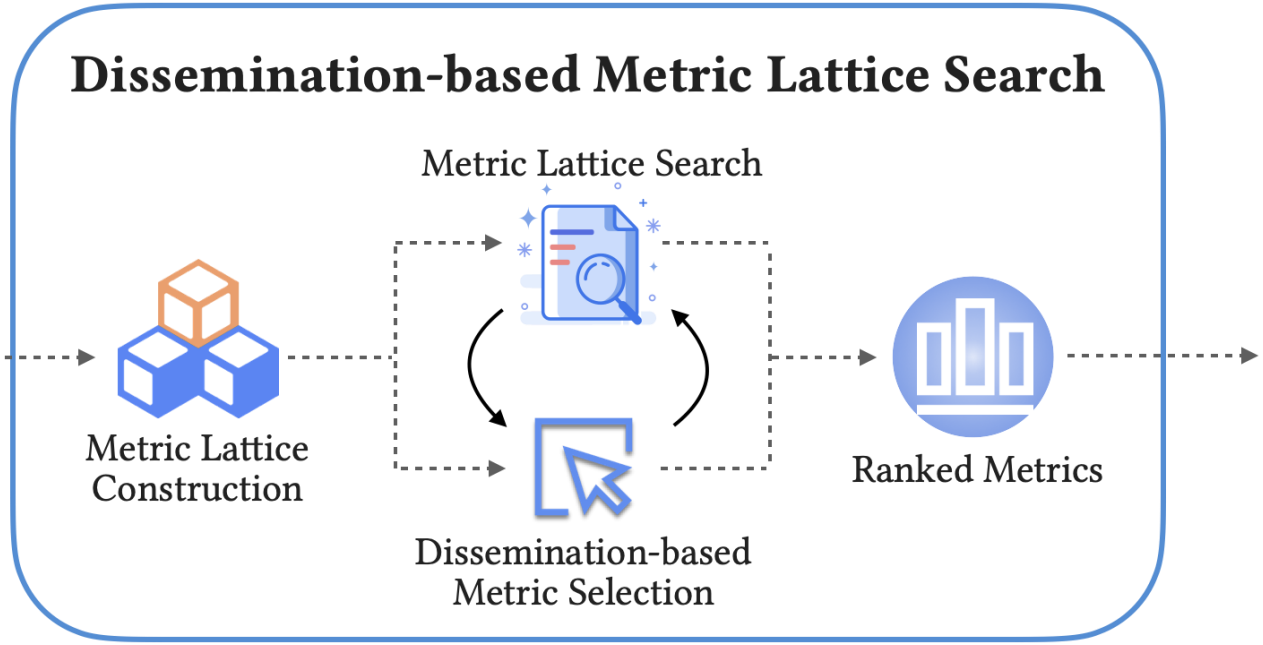
MicroRes: Degradation-based Metric Lattice Search



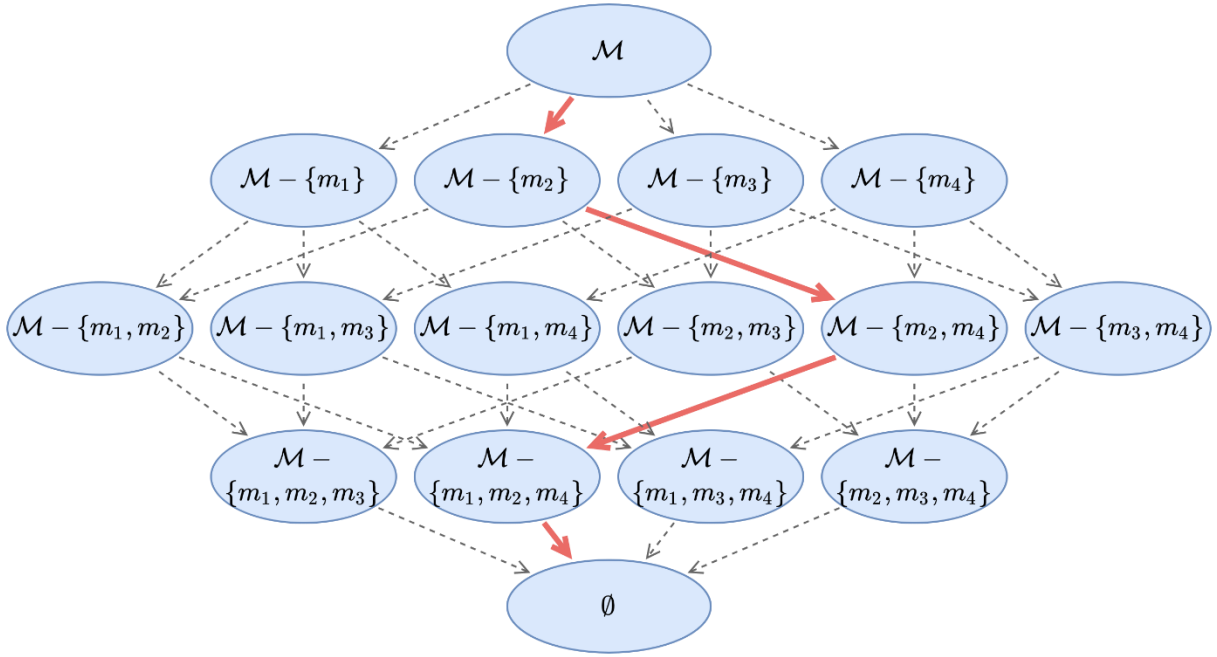
- Construct the metric lattice from the power set of $\boldsymbol{M}$.
  - Each node is a subset of $\boldsymbol{M}$.
  - Ordered by the subset-superset relation.



MicroRes: Degradation-based Metric Lattice Search

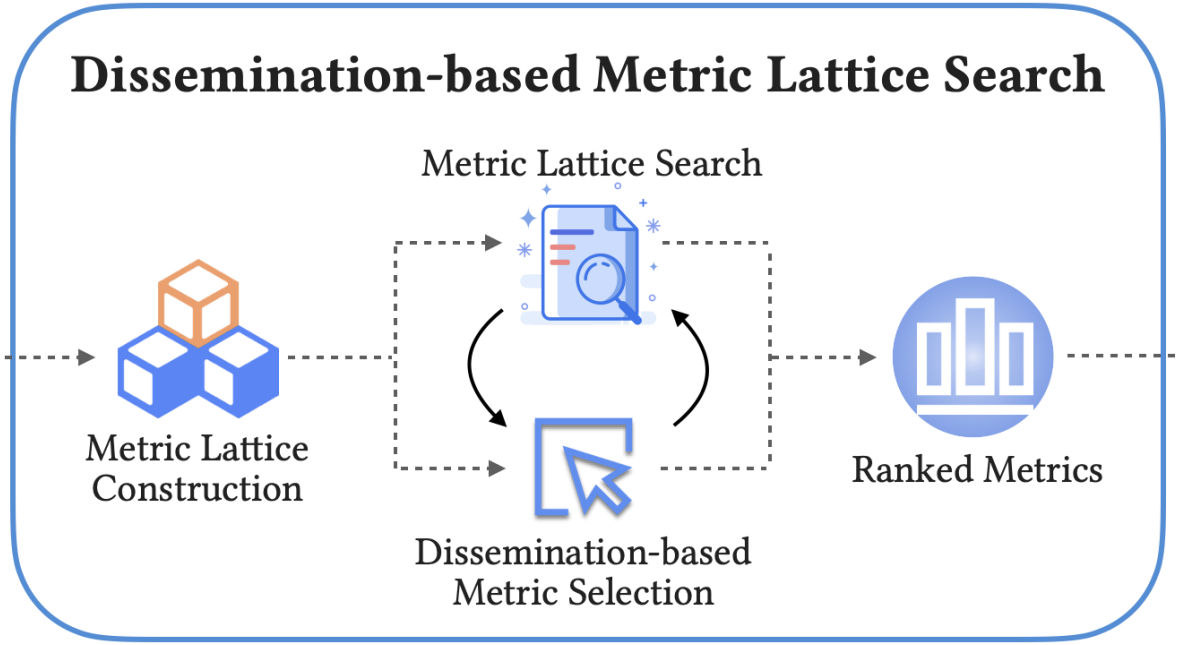


- Idea
  - Depth-first search from the upmost node to the bottommost node.
  - Select the metric that contributes most to the overall service degradation.



Please check the detailed algorithm in the paper.

MicroRes: Degradation-based Metric Lattice Search



Algorithm 2: Dissemination-based Metric Selection

Input: The monitoring metric subset $\mathcal{M}'$; The monitoring metrics during the failure injection period $\mathcal{M}'^{f}$; The monitoring metrics during the failure clearance period $\mathcal{M}'^{n}$

Output: The metric $m_i \in \mathcal{M}'$ where m_i contribute most to the overall service degradation

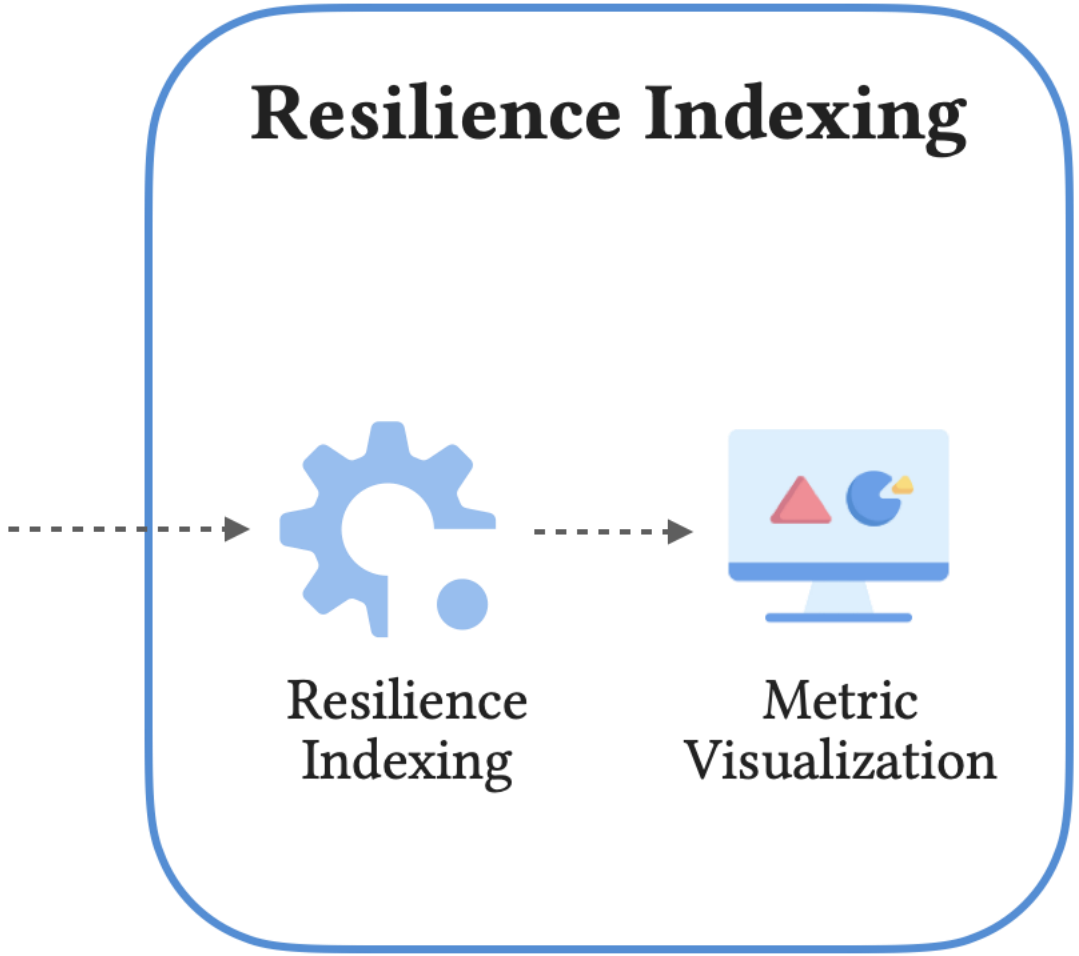
1 **Function** MetricSelection($\mathcal{M}'$, $\mathcal{M}'^{f}$, $\mathcal{M}'^{n}$):
2 T = length of the monitoring metrics
3 D = []
4 **for** $m_i \in \mathcal{M}'$ **do**
5 // Compute the performance difference of each individual metric
6 **for** $t = 1 \ldots T$ **do**
7 $\delta_i(t) = |m_i^f(t) - m_i^n(t)|$
8 **end**
9 $\hat{\delta}_i = \delta_i - \bar{\delta}_i$ // Normalize δ_i
10 D = [D; $\hat{\delta}_i$] // Concatenate the normalized performance difference
11 **end**
12 δ_{PC1} = PCA(D, $dim = 1$) // Reduce to one dimension via Principal Component Analysis
13 // Select the metric that contributes most to the performance difference
14 **for** $\hat{\delta}_i \in$ D **do**
15 c_i = Contribution($\delta_{PC1}, \hat{\delta}_i$)
16 **end**
17 $cmax = \max(c_i)$
18 $imax = \arg\max_i(c_i)$
19 **return** $cmax$, m_{imax}
20 **End**

Compute performance degradation

Select the metric with highest contribution

Please check the detailed algorithm in the paper.

MicroRes: Resilience Indexing



- Idea
  - If the degradation of system performance metrics cannot **propagate** to the degradation of user-aware metrics, the resilience is higher.
- Approach
  - Calculate the degradation contributed by $\boldsymbol{U}$ and $\boldsymbol{P}$.

$$D_{\mathcal{P}} = \sum_{m_i \in \mathcal{P}} \frac{c_i}{\log_2(rank(m_i; \mathcal{L}) + 1)}$$

$$D_{\mathcal{U}} = \sum_{m_i \in \mathcal{U}} \frac{c_i}{\log_2(rank(m_i; \mathcal{L}) + 1)}$$

  - Calculate the propagation.

$$r = \frac{1}{1 + e^{D_{\mathcal{U}} - D_{\mathcal{P}}}}$$

Experiment Settings

Table 3: Dataset Statistics

Dataset	$\|\mathcal{U}\|$	$\|\mathcal{P}\|$	#Microservices	#Failures	Failure Duration
Train-Ticket	30	195	15	24	10 minutes
Social-Network	50	325	25	10	5 minutes
Industry	2	12	(Undisclosed)	28	20 minutes

- Manual labeling of resilience
  - **PASS/FAIL** Done by two PhD students.
  - Verified by experienced engineers of Huawei.

- Evaluation Metrics
  - Mean Absolute Error (MAE)
  - Root Mean Squared Error (RMSE)
  - Cross Entropy (CE)
  - Accuracy
  - F1-score

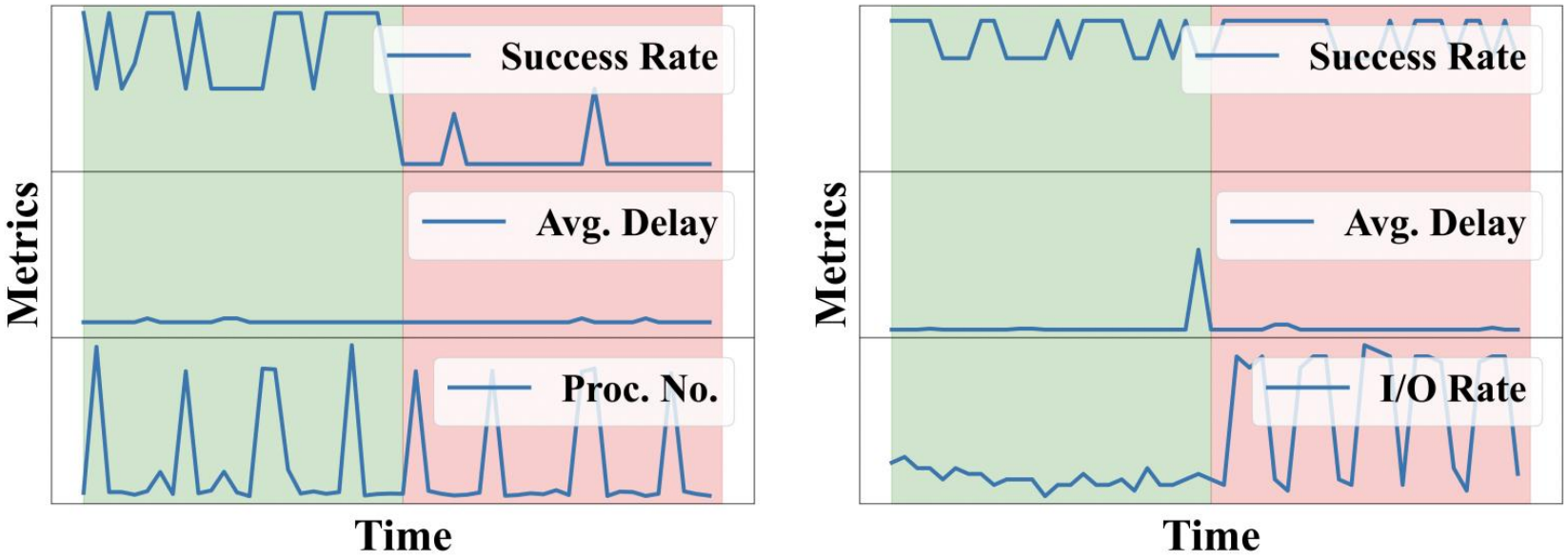
Effectiveness & Ablation Study

Table 2: Effectiveness Comparison (RQ1) and Ablation Study (RQ2) of MicroRes

Category	Method	*Train-Ticket*					*Social-Network*					*Industry*				
		CE	*MAE*	*RMSE*	*Acc*	*F1*	*CE*	*MAE*	*RMSE*	*Acc*	*F1*	*CE*	*MAE*	*RMSE*	*Acc*	*F1*
RQ1	SVC	0.8830	0.3497	0.5267	0.5802	0.7018	1.2608	0.3908	0.5657	0.5278	0.6383	0.6743	0.3786	0.4627	0.6786	0.7273
	RF	0.9399	0.3507	0.5277	0.5802	0.7018	0.6708	0.2358	0.4063	0.5833	0.6809	0.7477	0.4012	0.4865	0.5000	0.5882
	ET	0.8163	0.2999	0.4771	0.5926	0.7227	0.9160	0.3135	0.4927	0.6111	0.6818	0.5340	0.3100	0.3814	0.5714	0.6842
RQ2	MicroRes-euc	0.4464	0.1868	0.3384	0.6543	0.7846	0.7199	0.2861	0.4640	0.6389	0.7451	0.4409	0.3036	0.3729	0.6071	0.7027
	MicroRes-corr	0.3629	0.1730	0.3174	0.6914	0.8092	0.5969	0.2201	0.3865	0.6111	0.7407	0.4049	0.2882	0.3516	0.5714	0.6842
	MicroRes-cid	0.3725	0.1645	0.3037	0.8148	0.8966	0.5154	0.1851	0.3326	0.8333	0.9091	0.3855	0.2737	0.3304	0.8571	0.9130
	MicroRes	**0.3246**	**0.1618**	**0.2993**	**0.9012**	**0.9481**	**0.3766**	**0.1814**	**0.3382**	**0.8611**	**0.9231**	**0.2977**	**0.2436**	**0.2812**	**0.8929**	**0.9362**

MicroRes achieves the best performance on all the datasets in terms of all evaluation metrics.

Successful Cases



(a) Case 1: Process Killed **(FAIL)** (b) Case 2: High I/O Rate **(PASS)**

Figure 4: Two successful cases in the industrial dataset. The green area means the normal period and the red area means the failure injection period.

ARISE
Automated Reliable Intelligent Software Engineering

香港中文大學
The Chinese University of Hong Kong

HUAWEI | Huawei Cloud

Thank You!