ARISE
Automated Reliable Intelligent Software Engineering

香港中文大學
The Chinese University of Hong Kong

Intelligent Reliability Management in Large-scale Cloud Systems

Jinyang Liu

PhD Oral Defense

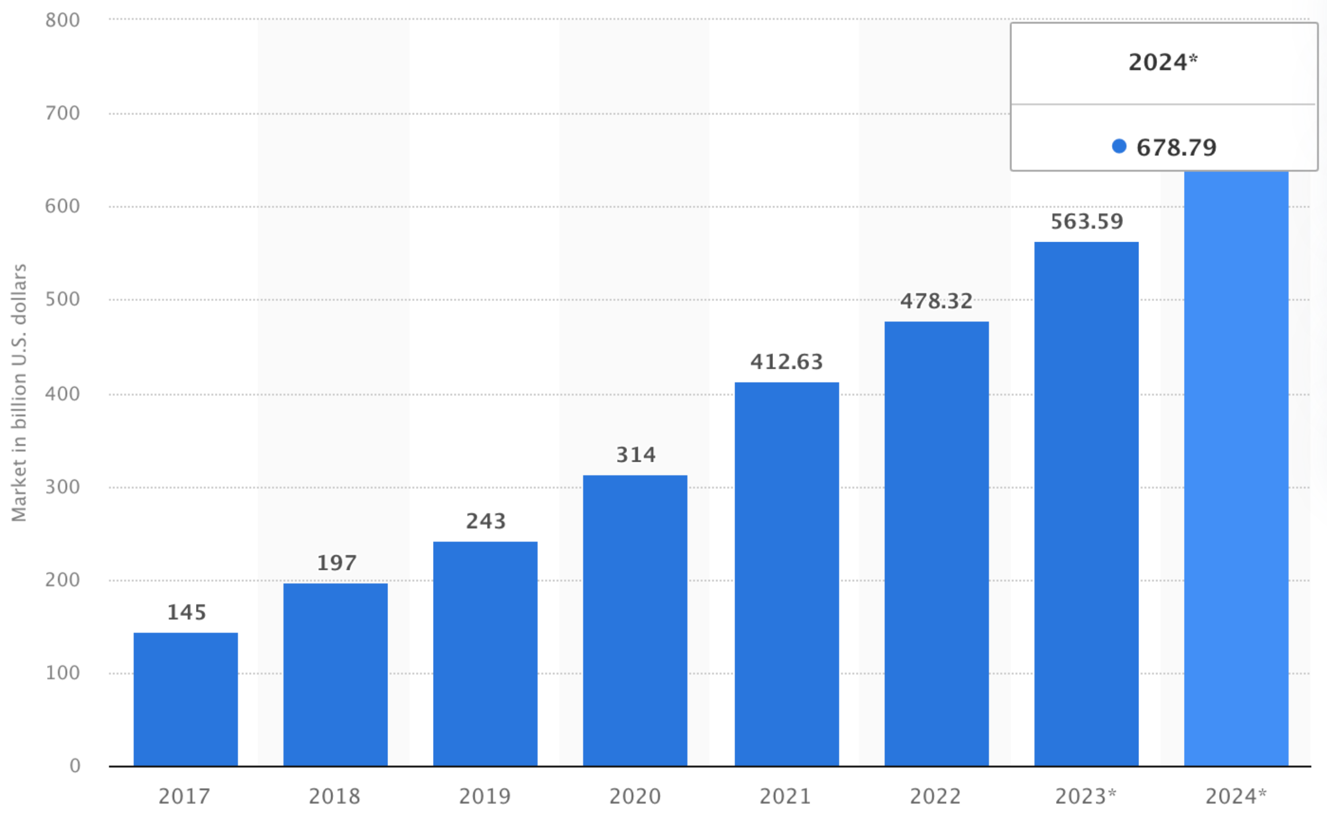
Supervisor: Prof. Michael R. Lyu

June 24, 2024

Background

Many **essential** applications have migrated to the cloud.

The public cloud market is increasing, estimated **679 billion U.S. dollars** in 2024.



https://www.statista.com/statistics/273818/global-revenue-generated-with-cloud-computing-since-2009/

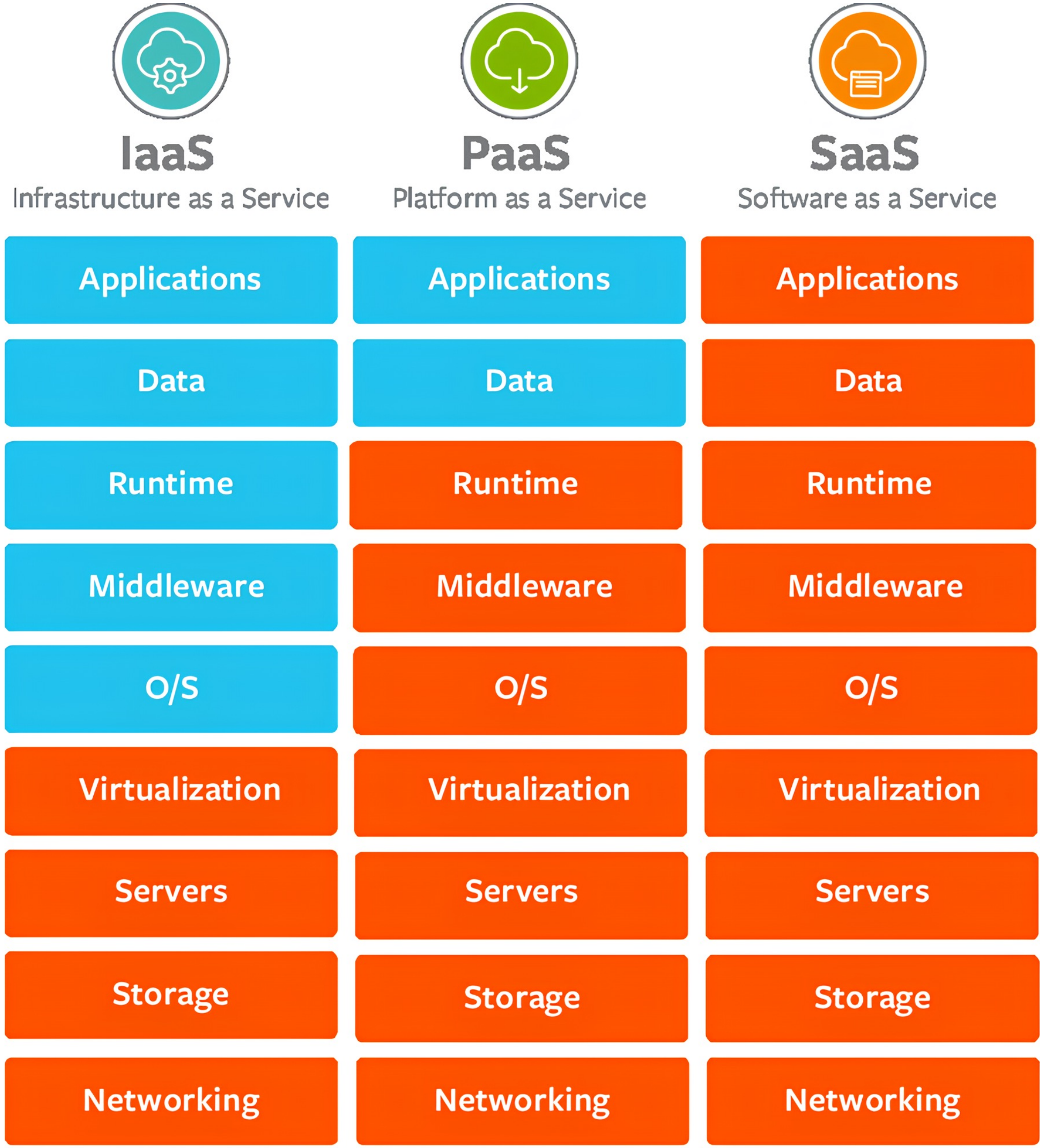
Background

Cloud Providers

- Deliver services in different layers of virtualization and abstraction
- Maintain most resources

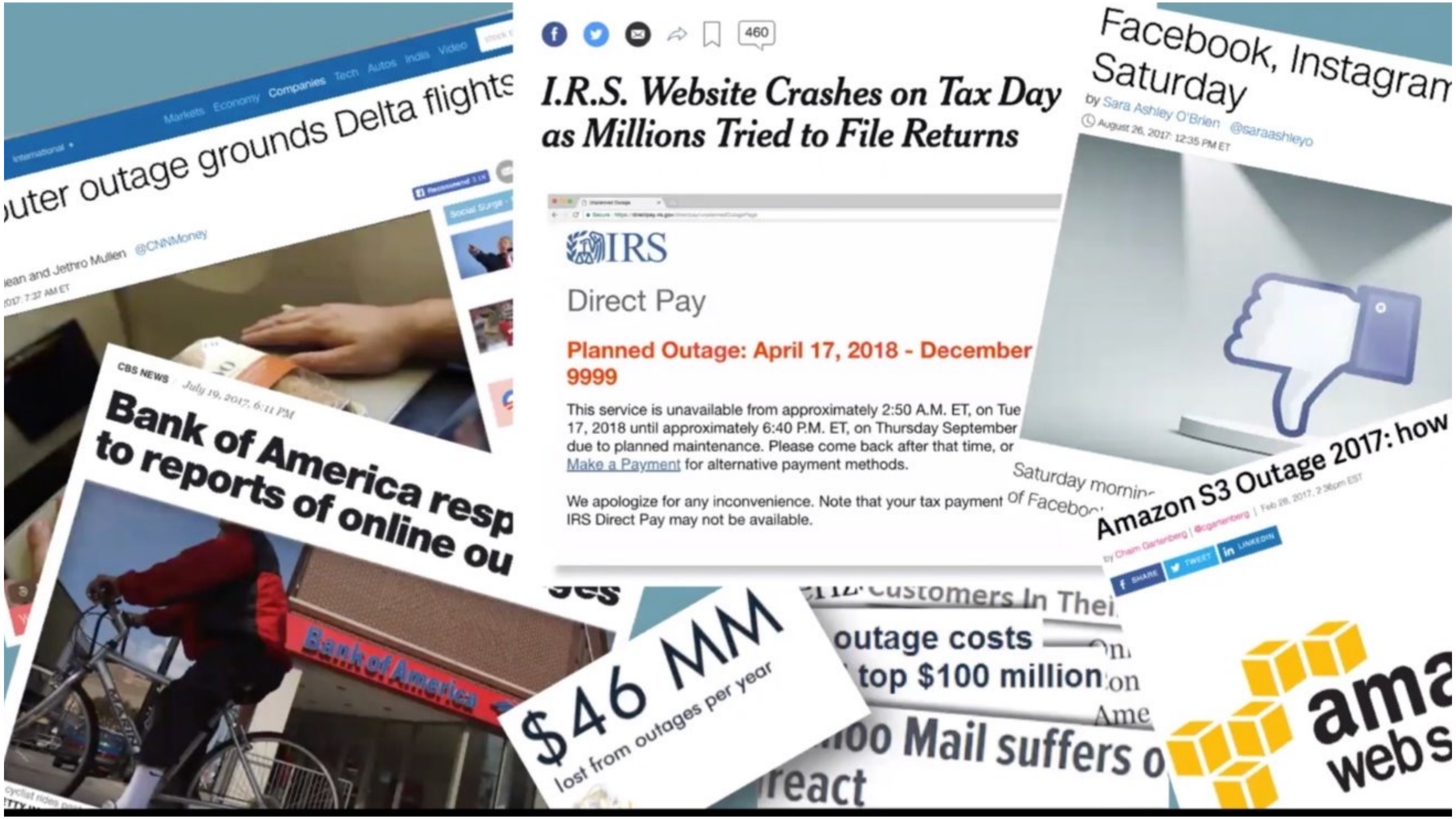
Customers

- Consume and utilize the services
- Little maintenance effort



Background

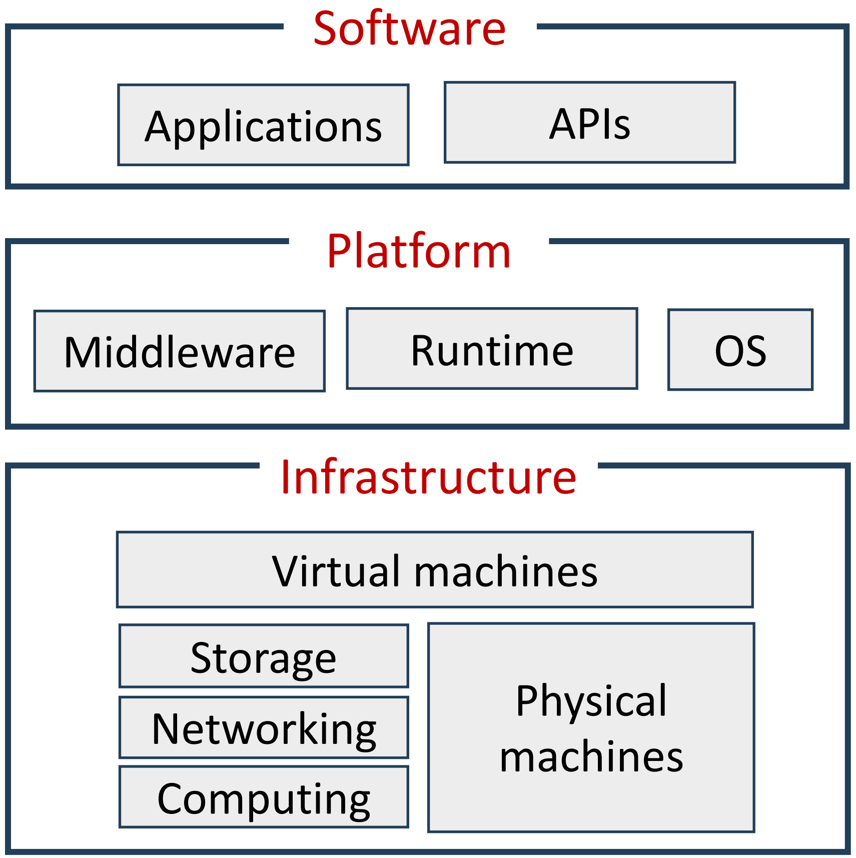
However, incidents can interrupt these services…

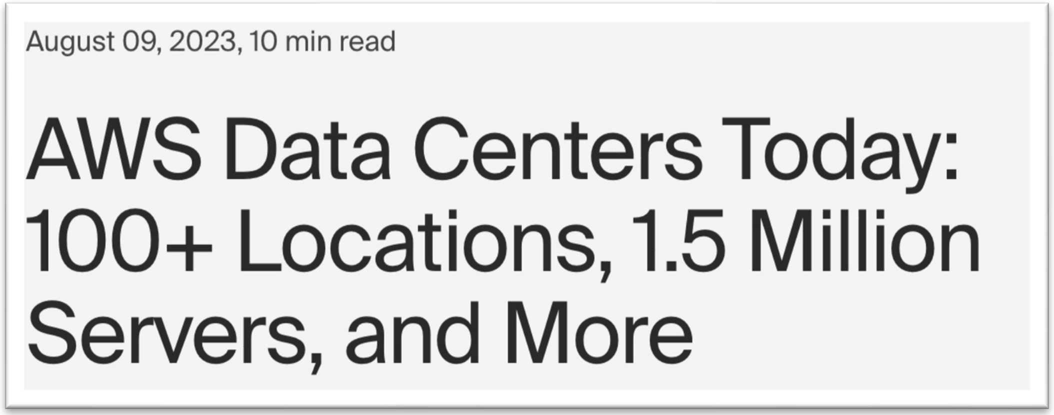


It is crucial yet challenging to
ensure reliability of cloud systems!

Background

- Challenge 1: Large scale of cloud systems



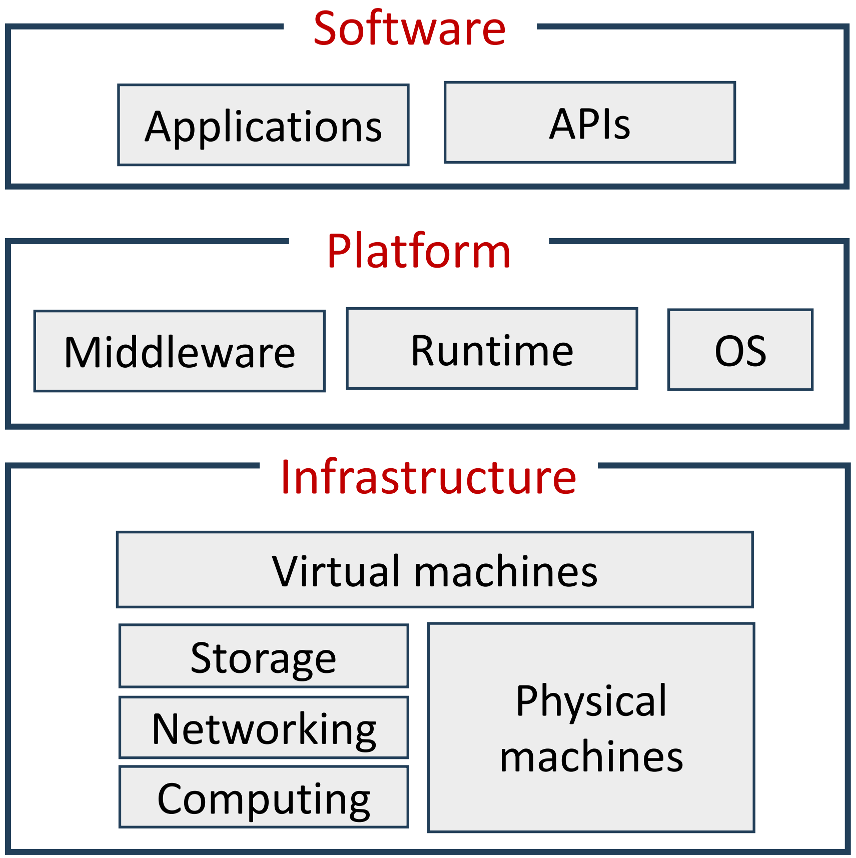

August 09, 2023, 10 min read

AWS Data Centers Today: 100+ Locations, 1.5 Million Servers, and More

[AWS Data Centers Today: 100+ Locations, 1.5 Million Servers, and More (cloudzero.com)]
[AWS brings economic benefits to California, Ohio, Oregon, Virginia (aboutamazon.com)]

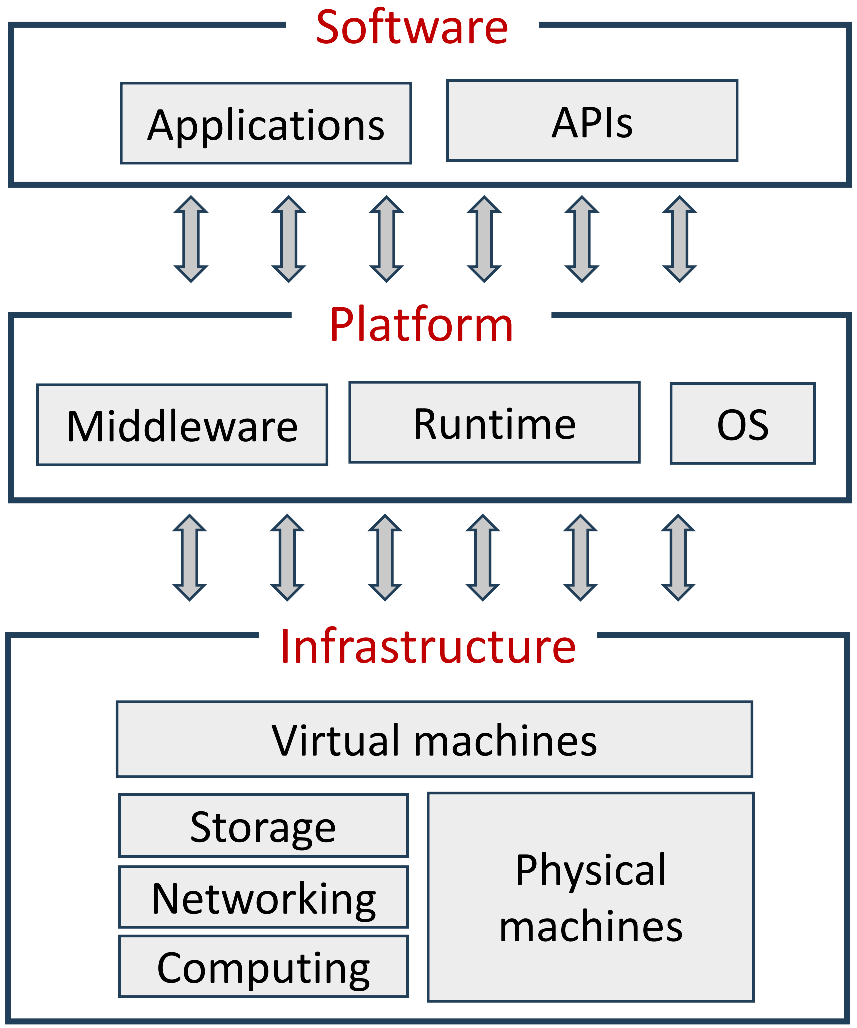
Background

- Challenge 2: Complicated dependencies between services



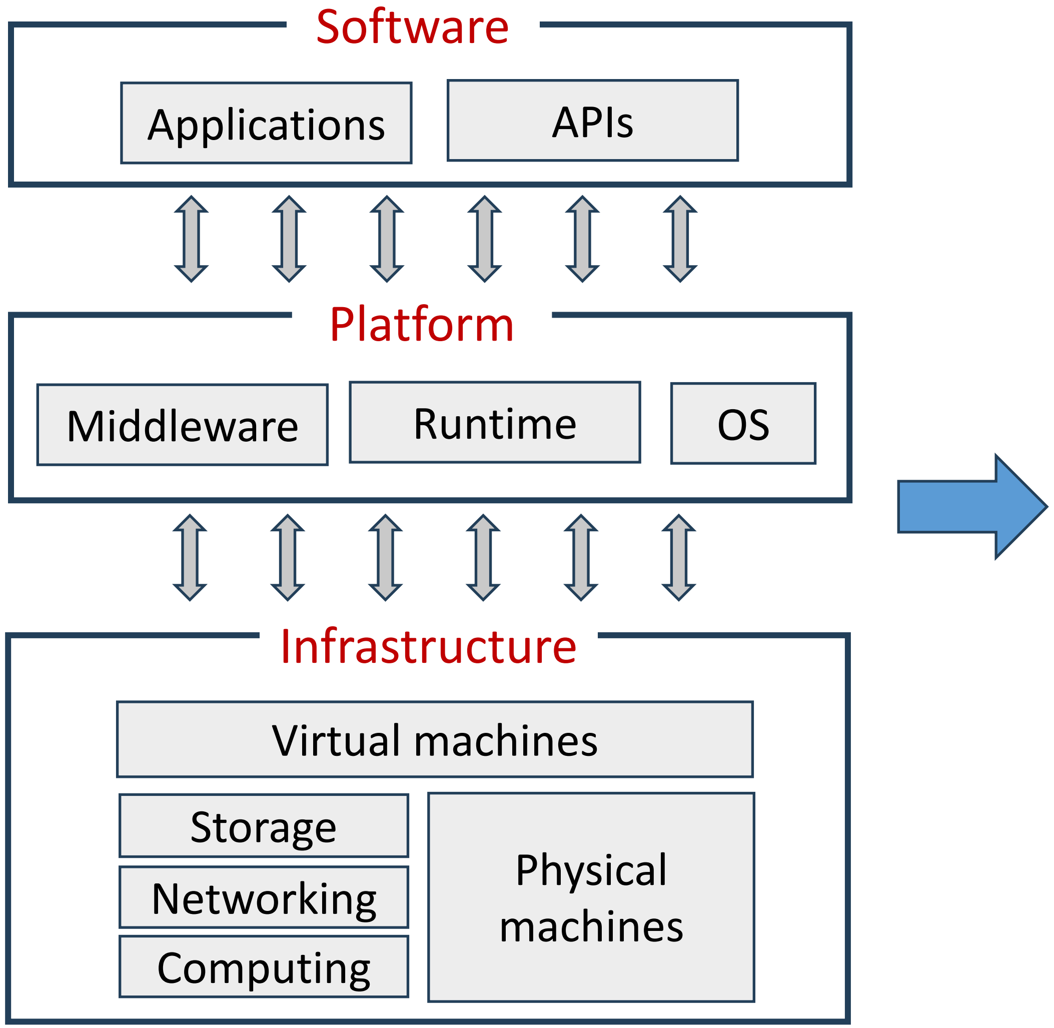
Background

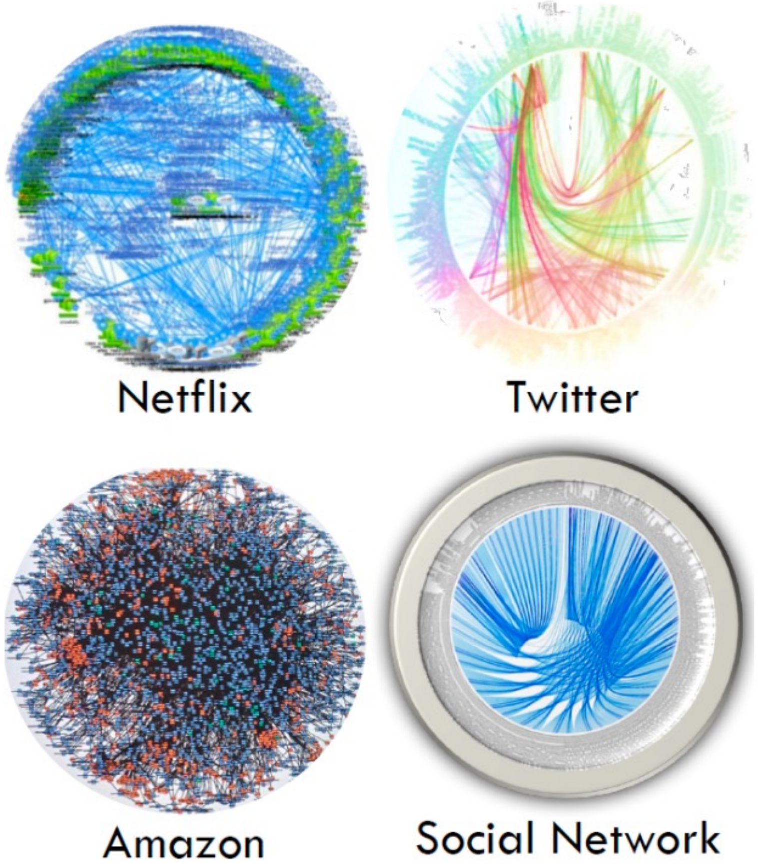
- Challenge 2: Complicated dependencies between services



Background

- Challenge 2: Complicated dependencies between services



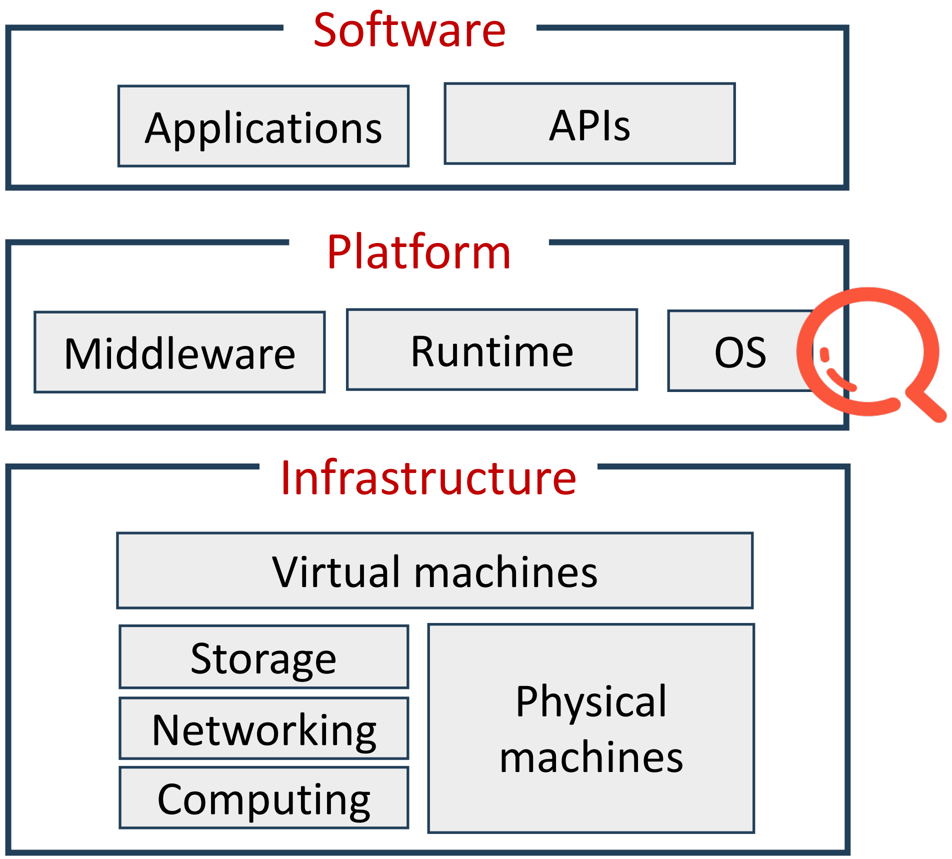


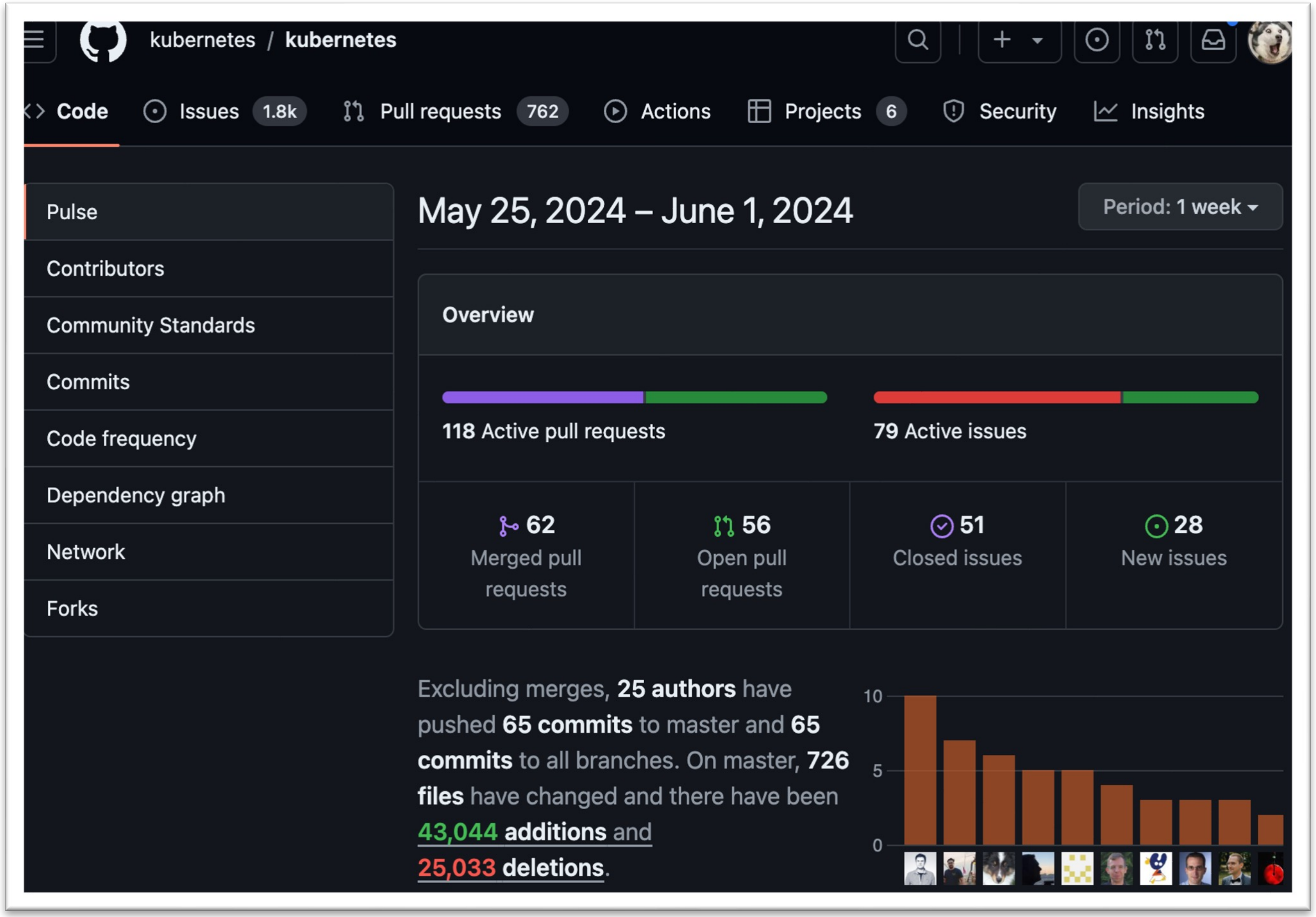
Dependencies between microservices

Zhang, Y., Gan, Y., & Delimitrou, C. (2019). uqSim: Scalable and Validated Simulation of Cloud Microservices. arXiv preprint arXiv:1911.02122.

Background

- Challenge 3: Evolving nature of cloud software

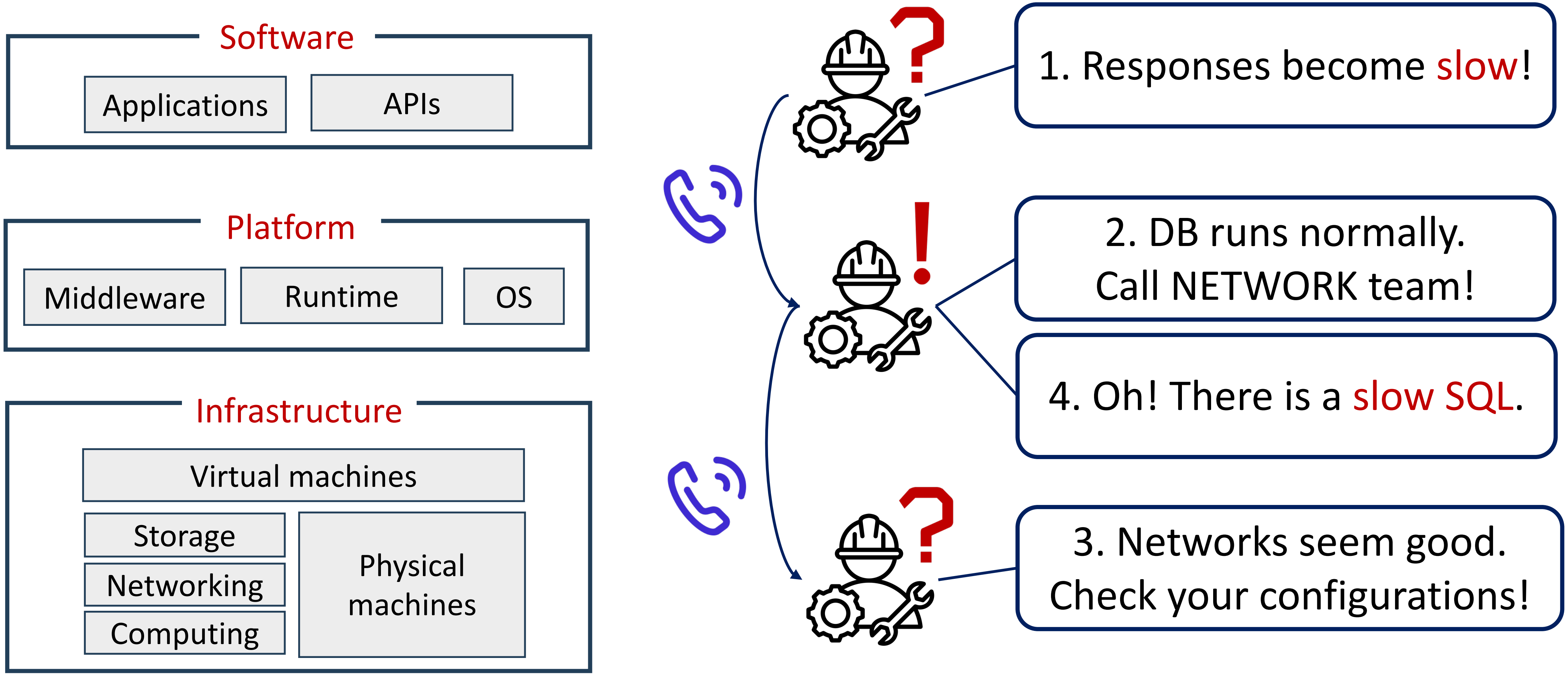




One week: 726 file changes,
43,044 additions, 25,033 deletions

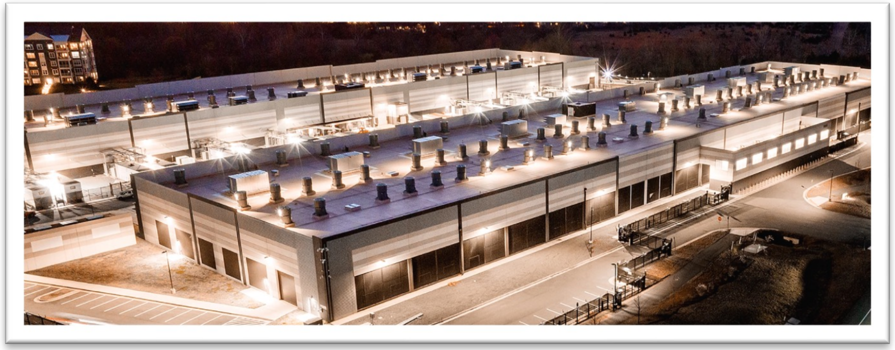
Background

- Challenge 4: Limited observations of cloud systems

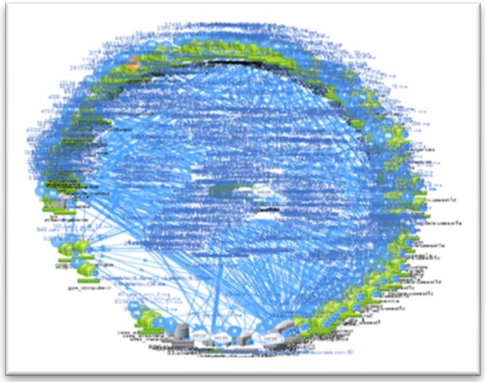


Background

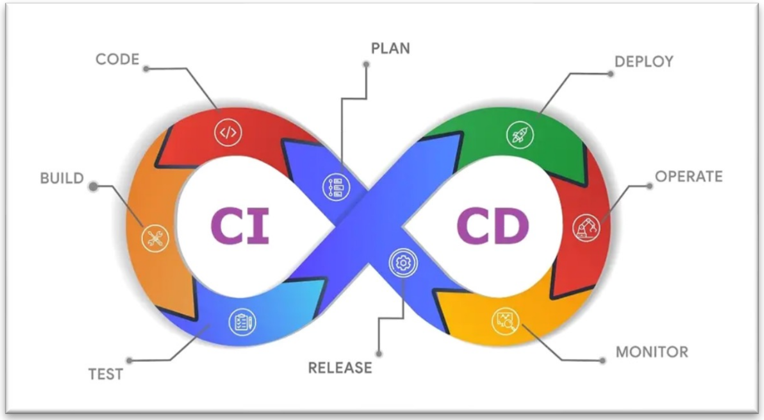
- Challenges in ensuring reliability of cloud systems



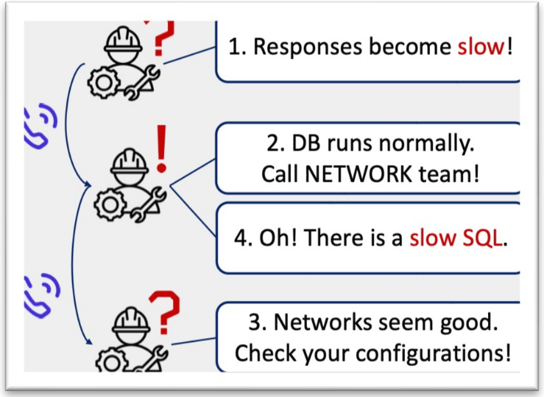
① Large Scale



② Complicated Dependencies

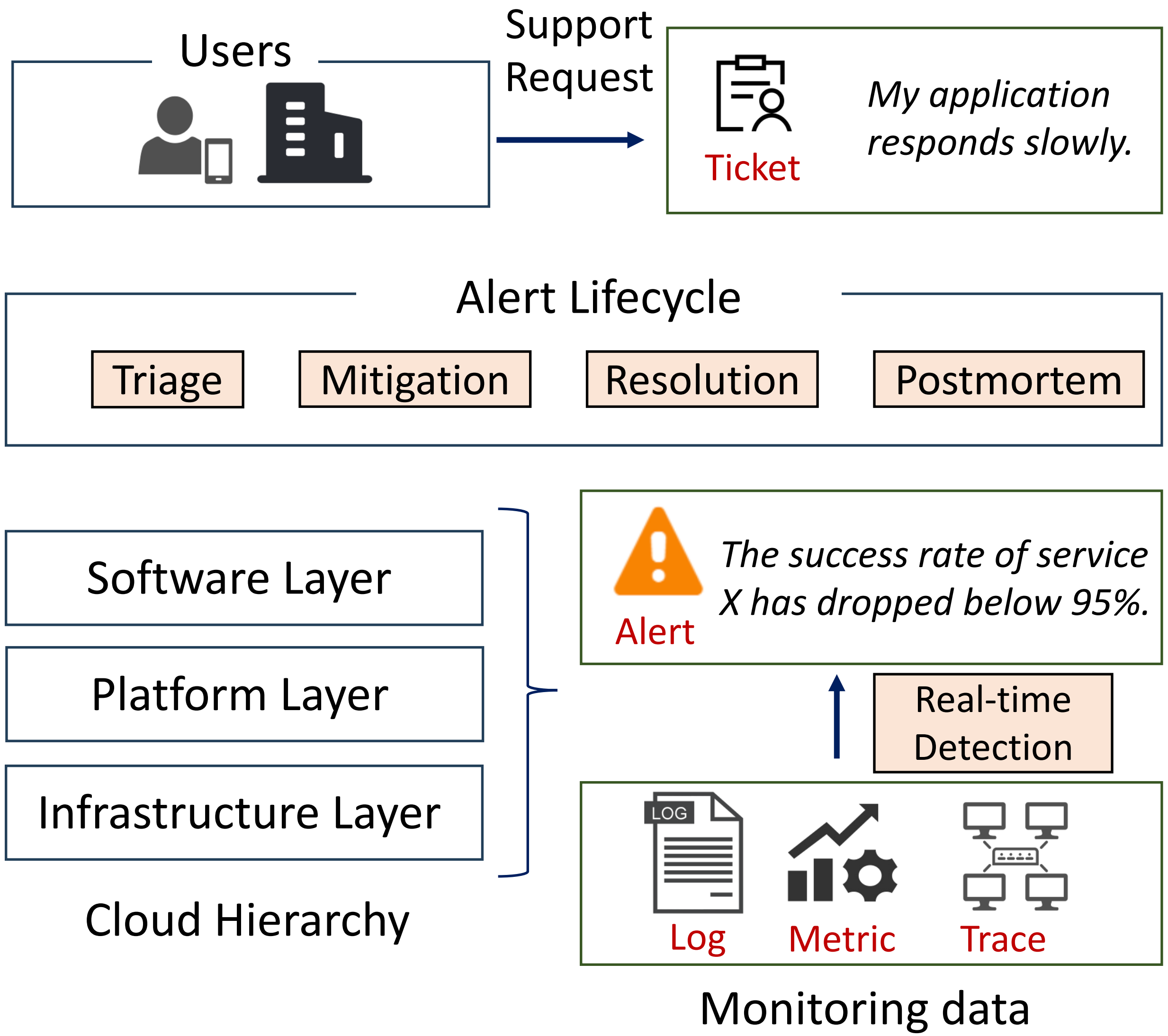


③ Fast Evolving



④ Limited Observations

Background: reliability management of modern cloud systems



On-call Engineers (OCE)

When an incident (an unexpected problem) happens:

① Called by alerts (Triage)

② Understand alerts

③ Check monitoring data

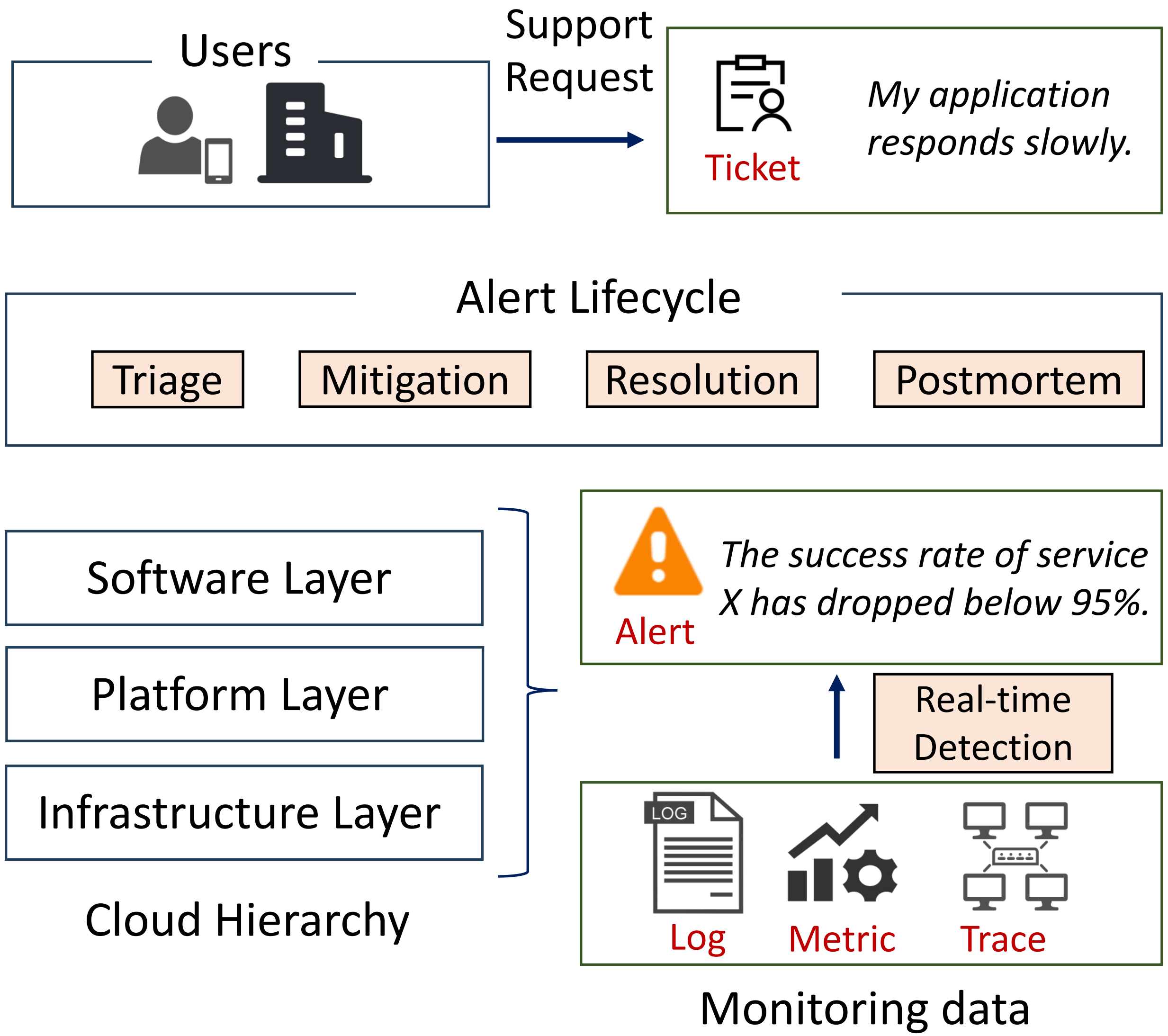
④ Diagnosis & Recovery

(Mitigation)

⑤ Fix bugs (Resolution)

⑥ Summarize (Postmortem)

Background: reliability management of modern cloud systems



On-call Engineers (OCE)

- ⊗ Labor-intensive
- ⊗ Skill-intensive
- ⊗ Error-prone

When an incident (an unexpected problem) happens:

① Called by alerts (Triage)

② Understand alerts

③ Check monitoring data

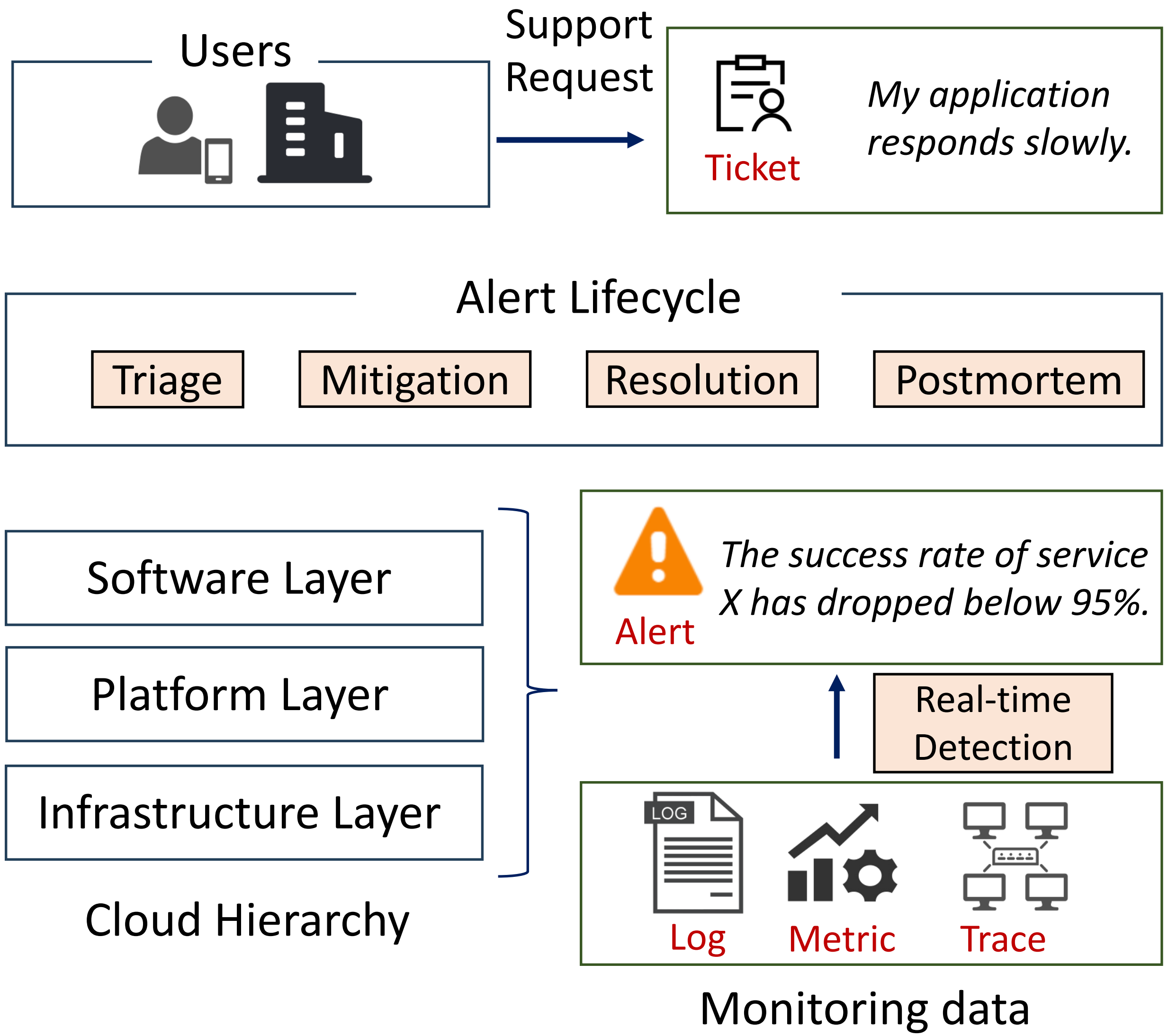
④ Diagnosis & Recovery

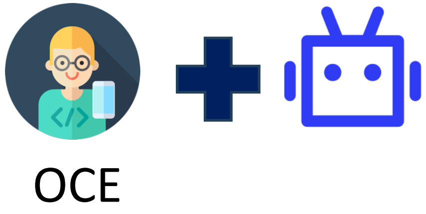
(Mitigation)

⑤ Fix bugs (Resolution)

⑥ Summarize (Postmortem)

Our goal: Intelligent reliability management





Intelligent solutions

- ✅ Automated
- ✅ Efficient
- ✅ Scalable
- ✅ Evolving

When an incident (an unexpected problem) happens:

① Called by alerts (Triage)

② Understand alerts

③ Check monitoring data

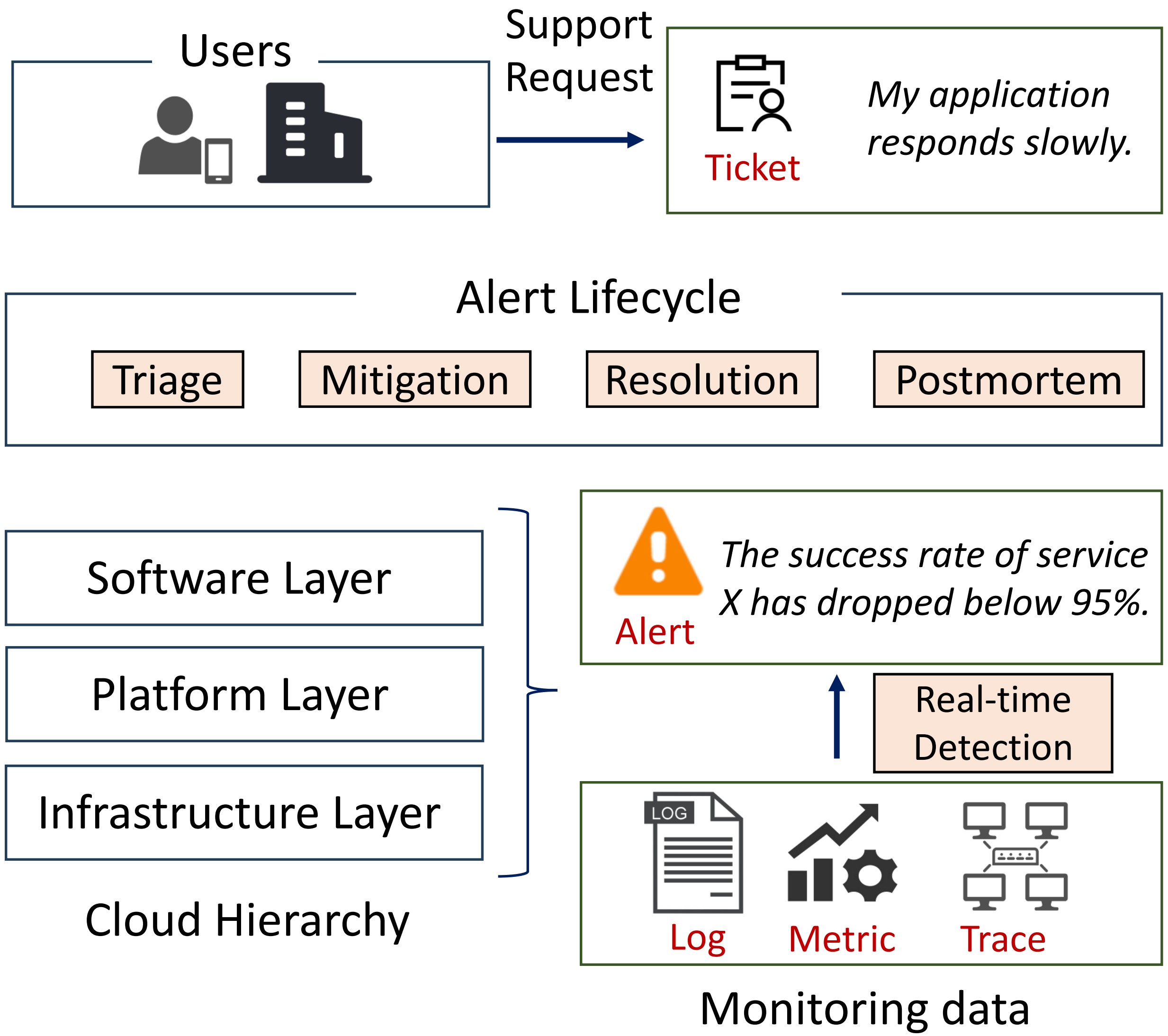
④ Diagnosis & Recovery

(Mitigation)

⑤ Fix bugs (Resolution)

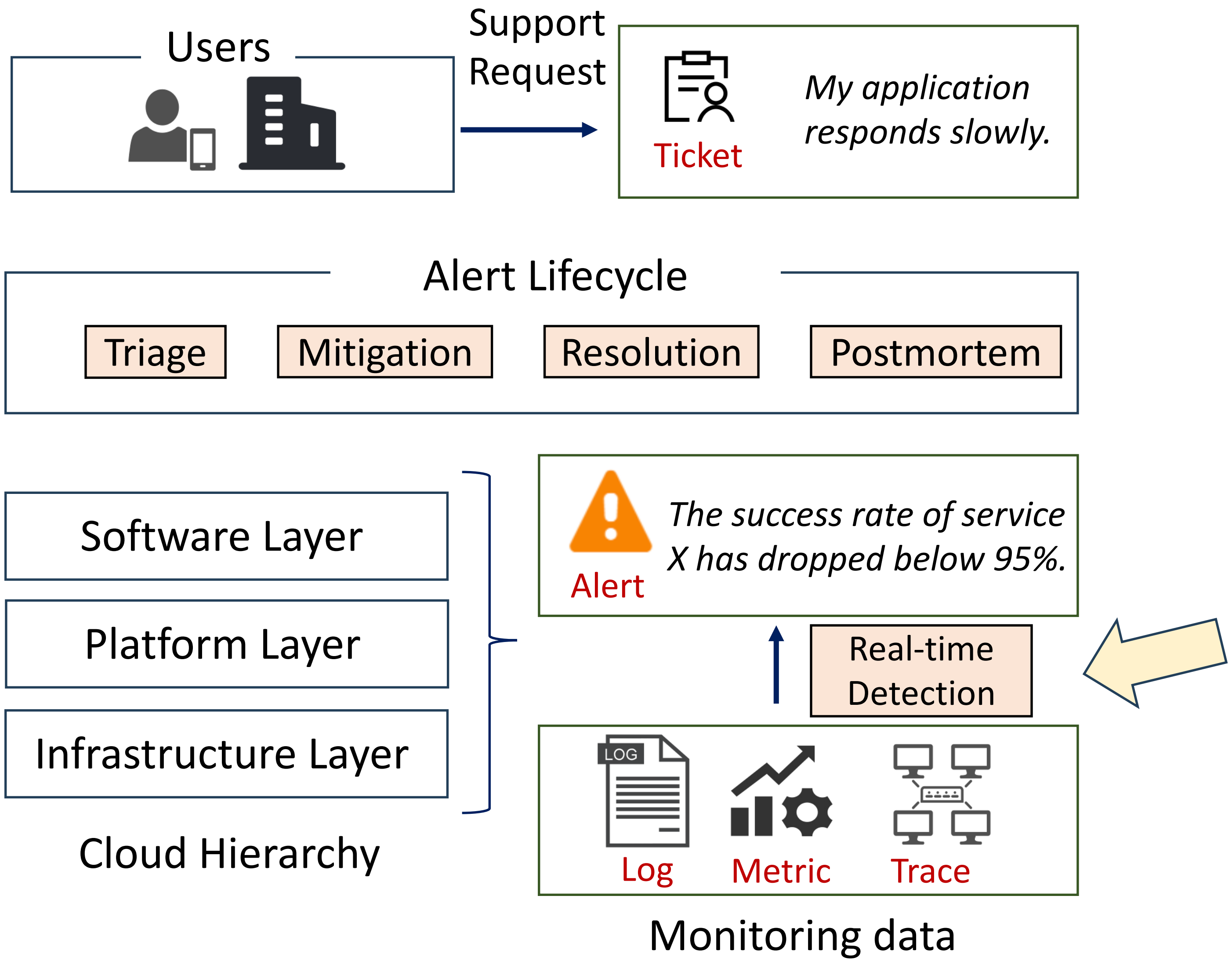
⑥ Summarize (Postmortem)

Our goal: Intelligent reliability management



Thesis Contribution

Our goal: Intelligent reliability management

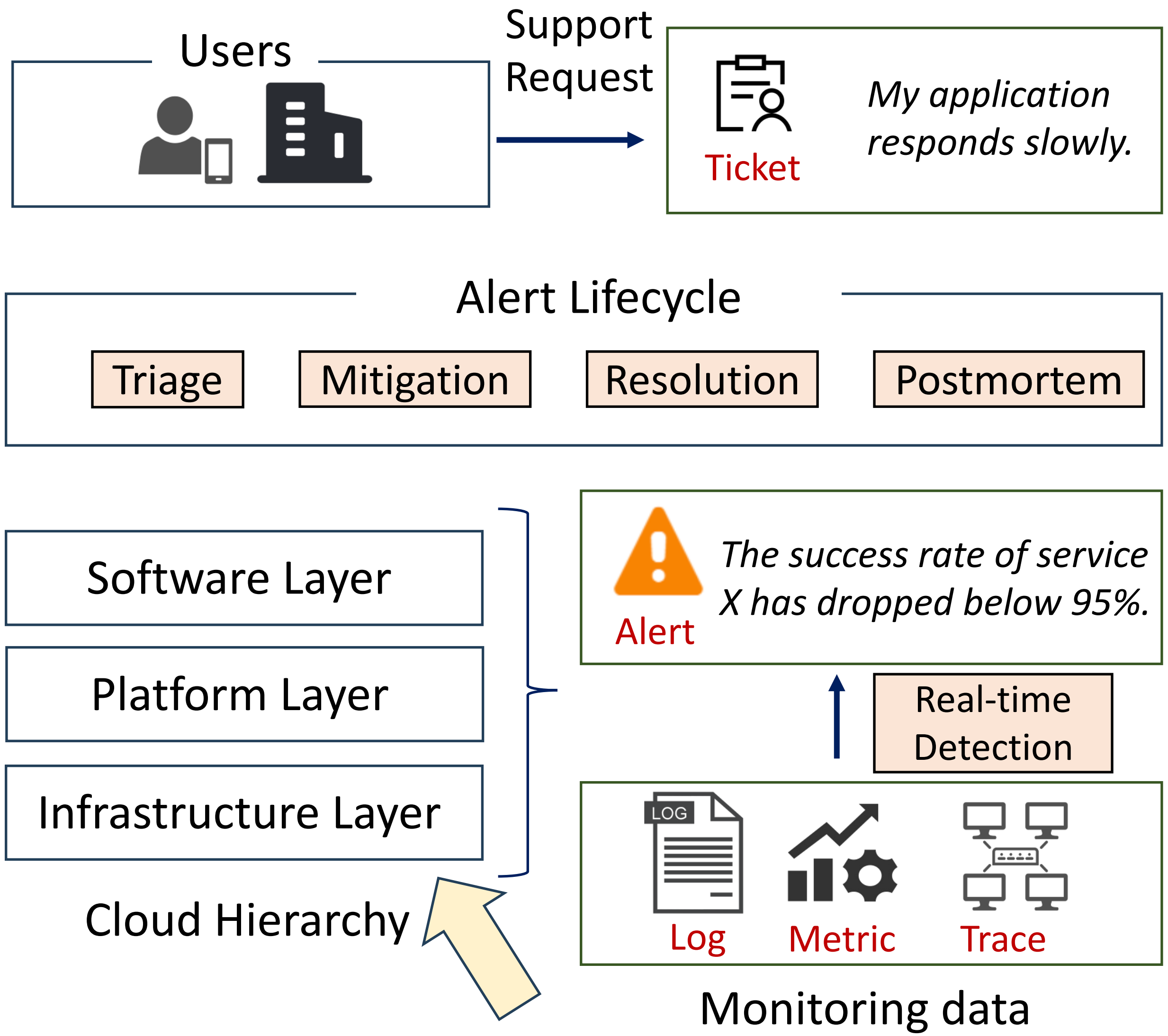


Thesis Contribution

① Sealog

Scalable and adaptive log-based anomaly detection

Our goal: Intelligent reliability management



Thesis Contribution

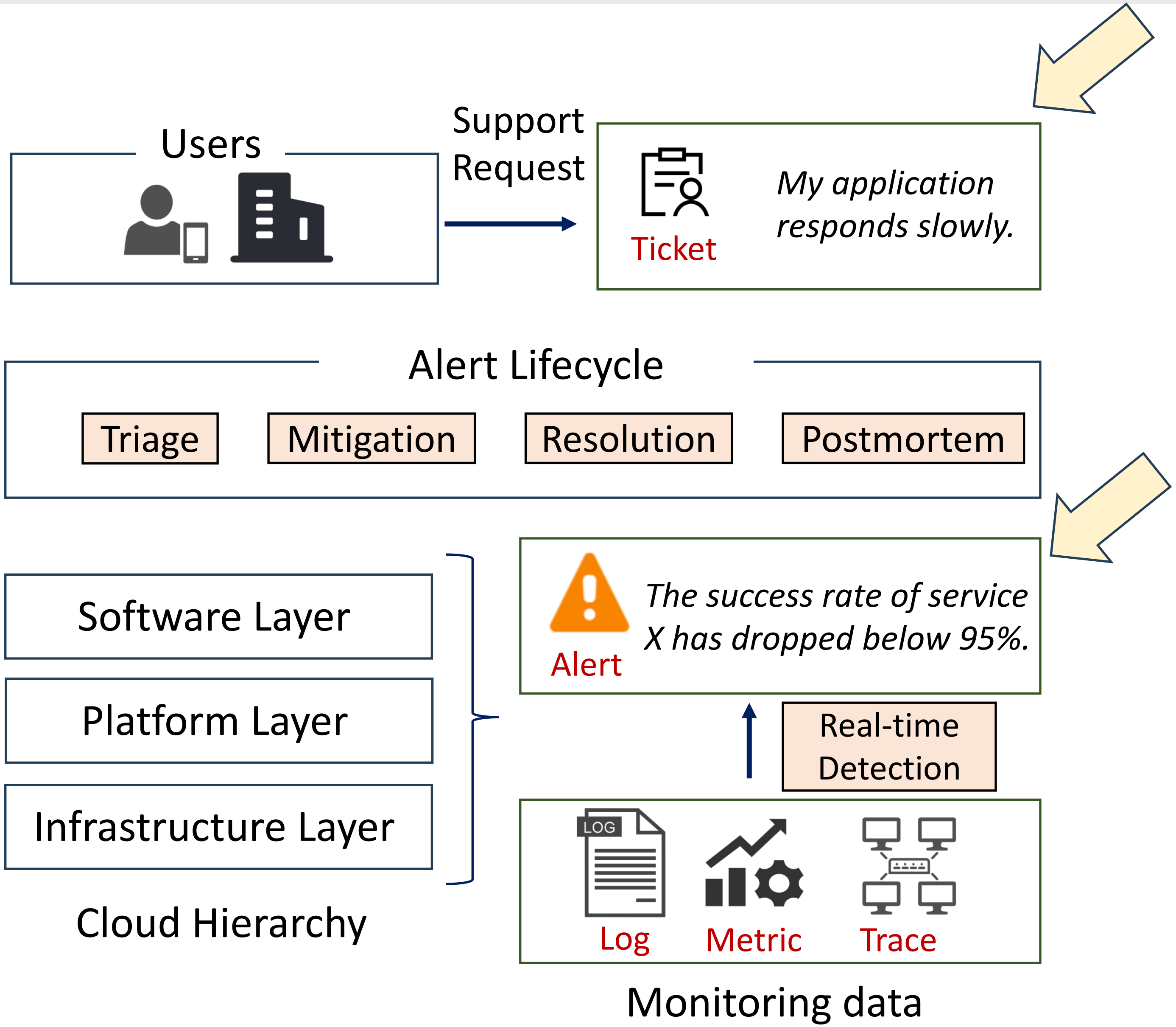
① Sealog

Scalable and adaptive log-based anomaly detection

② Prism

Improving observability across different layers of cloud hierarchy

Our goal: Intelligent reliability management



Thesis Contribution

① Sealog

Scalable and adaptive log-based anomaly detection

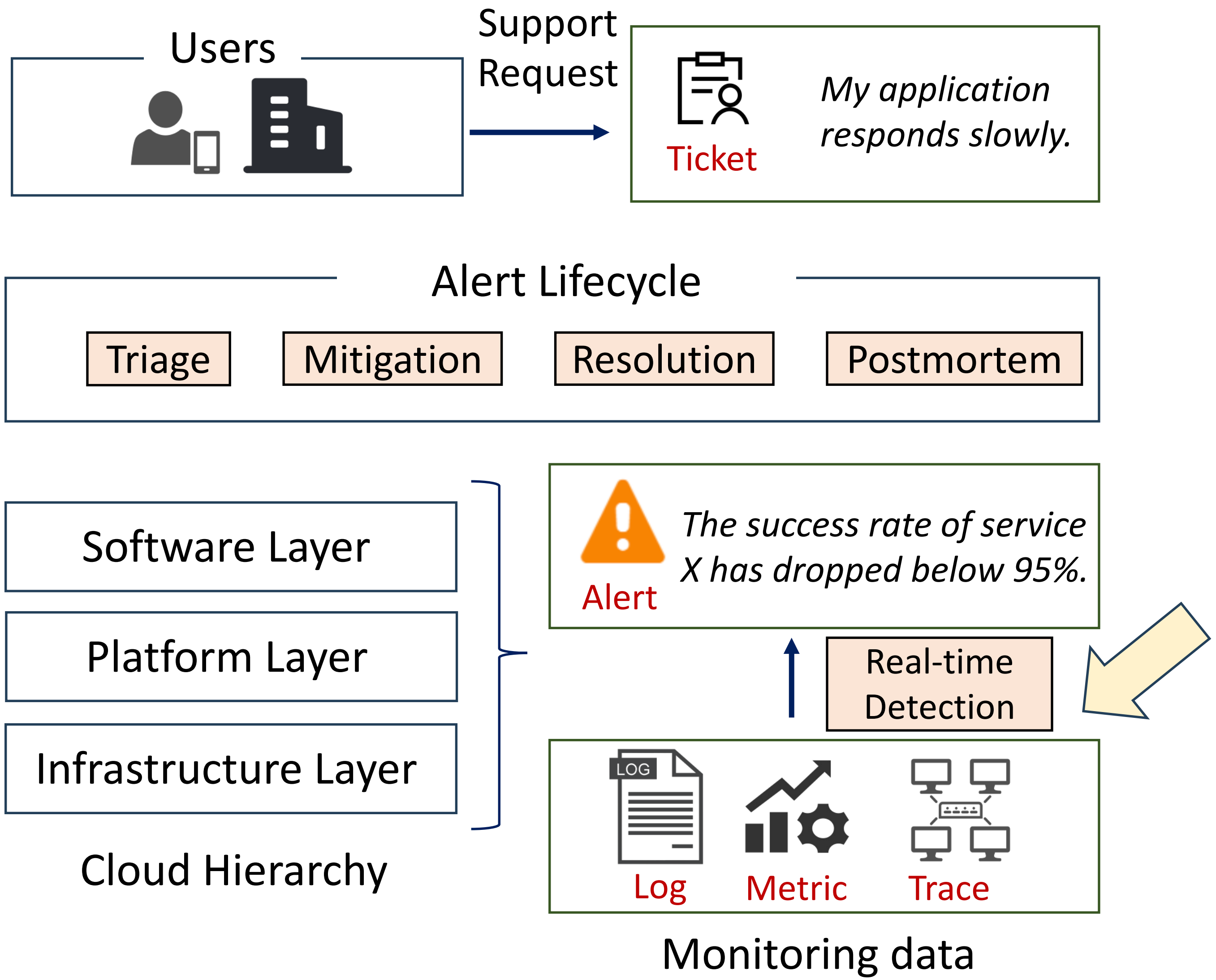
② Prism

Improving observability across different layers of cloud hierarchy

③ iPACK

Correlating tickets and alerts for more comprehensive ticket deduplication

Our goal: Intelligent reliability management



Thesis Contribution

① Sealog

Scalable and adaptive log-based anomaly detection

② Prism

Improving observability across different layers of cloud hierarchy

③ iPACK

Correlating tickets and alerts for more comprehensive ticket deduplication

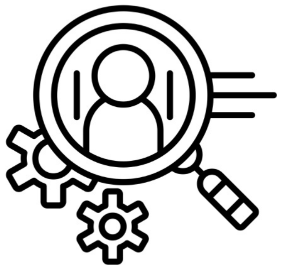
Background

- Log data is one of the most important data sources.

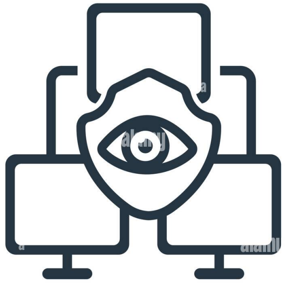
Software runtime behaivors

Trouble shooting

Performance analysis



User behavior analysis



Security audit

Background

- Automatic log-based anomaly detection is essential.

Anomaly: unexpected patterns or events that deviate from the norm or expected operation.

CreateInstance:

Get request..
Authenticating..
No enough resources..
Returning..

...

Example of a log anomaly

Focus: template anomalies

```
Sep 18 08:47:22 DEBUG Partition rdd_2_1 not found in 127.0.0.1
Sep 18 08:47:35 DEBUG Partition rdd_2_2 not found in 127.0.0.1
Sep 18 08:47:49 DEBUG Partition rdd_2_1 not found in 127.0.0.1
Sep 18 08:48:17 DEBUG Partition rdd_2_3 not found in 127.0.0.1
```

Raw logs

Log Parsing

Timestamp	Event	Level	Log Templates	Parameters
Sep 18 08:47:22	e1	DEBUG	Partition <*> not found in <*>	rdd_2_1 | 127.0.0.1
Sep 18 08:47:35	e1	DEBUG	Partition <*> not found in <*>	rdd_2_2 | 127.0.0.1
Sep 18 08:47:49	e1	DEBUG	Partition <*> not found in <*>	rdd_2_1 | 127.0.0.1
Sep 18 08:48:17	e1	DEBUG	Partition <*> not found in <*>	rdd_2_3 | 127.0.0.1

Structured logs

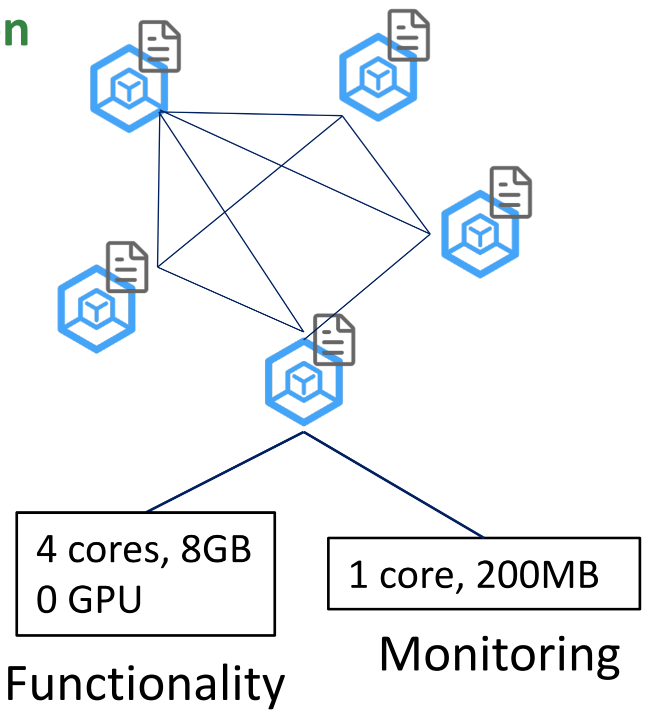
Motivation: a pilot study

- Characteristics of log data in real-world industrial environment

- Datasets
  - 20 microservices
  - Duration: 1 month
  - 1.48 million+ lines
  - 3,241 templates

HUAWEI CLOUD

Transmission overhead



Limited local resource

- Characteristic 1: **Massive and distributed**

- Requirement 1: **Lightweight** for local analysis

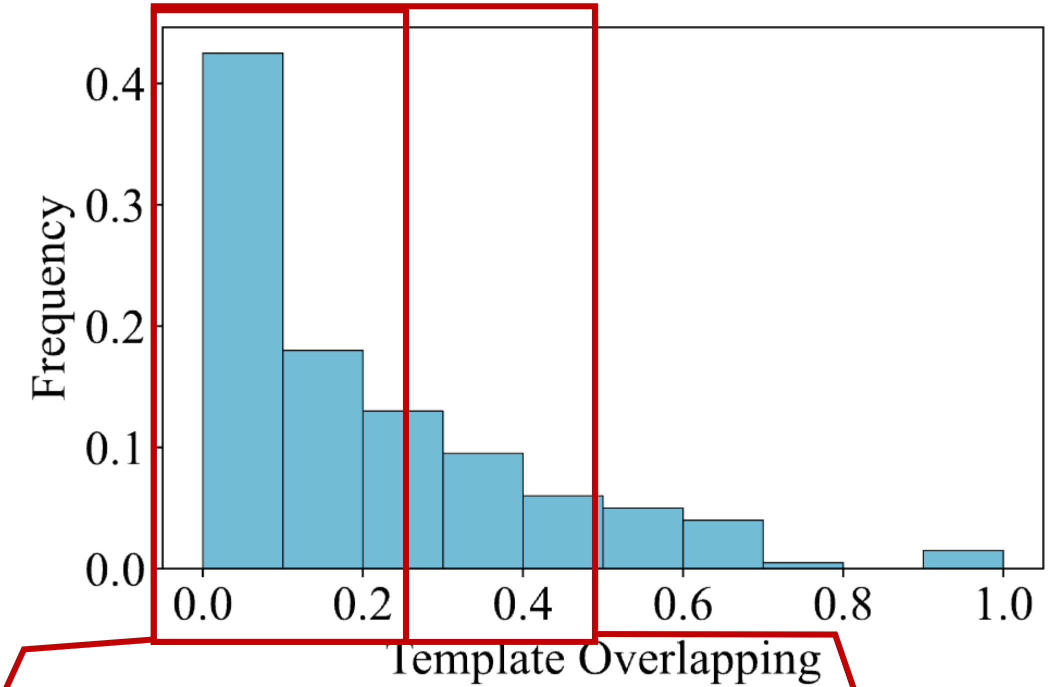
Motivation: a pilot study

- Characteristics of log data in real-world industrial environment

Share templates?

Comparison 1: Across-microservice

1. Pairwise combine 20 microservices
2. Compute their template set similarity: $\frac{S1 \cap S2}{S1 \cup S2}$



- 40% microservice pairs share no overlapping templates
- 80% microservices pairs share <50% templates overlapping

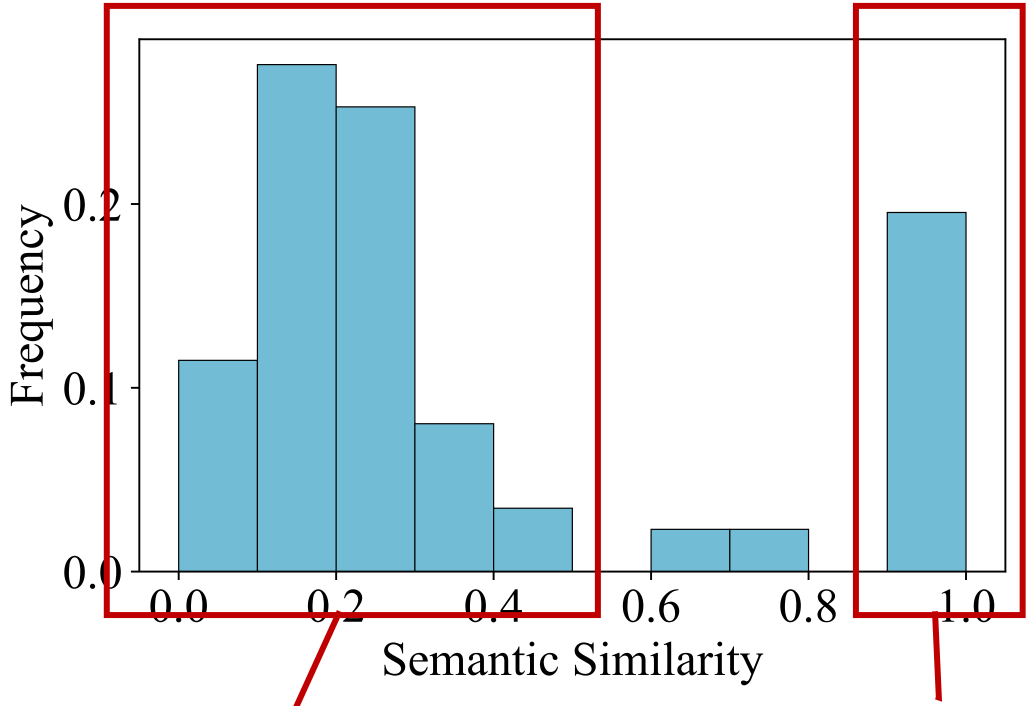
Motivation: a pilot study

- Characteristics of log data in real-world industrial environment

Share templates?

Comparison 2: Across-timeframe

1. Within each microservice
2. Represent log semantics with OpenAI-embedding
3. Compare log pairs before and after Feb 15 based on Cosine similarity



- Most new logs share little semantic similarities with seen logs
- Some logs may repeat over time.

Motivation: a pilot study

- Characteristics of log data in real-world industrial environment

Comparison 1: Across-microservice

- 40% microservices pair share no overlapping templates
- 80% microservices pairs share <50% templates overlapping

Comparison 2: Across-timeframe

- Some logs may repeat over time
- Most new logs share little semantic similarities with seen logs

- Characteristic 2:
  Diverse across different microservices

- Requirement 2:
  Accurate enough for various logs

- Characteristic 3:
  Evolving overtime

- Requirement 3:
  Adaptive to unseen logs

Motivation: a pilot study

- Existing solutions cannot fulfill all requirements

  - Requirement 1:
    Lightweight for local analysis
  - Requirement 2:
    Accurate enough for various logs
  - Requirement 3:
    Adaptive to unseen logs

Loglizer	*Statistics-based*
DeepLog LogAnomaly	*Without handling unseen cases*
RobustLog NeruaLog	*Label-intensive* *Compute-intensive*

[ICSE'22] Le V H, Zhang H. Log-based anomaly detection with deep learning: How far are we?[C]//Proceedings of the 44th international conference on software engineering. 2022: 1356-1367.

Key idea

- Integrating large language models (LLM) with lightweight ML methods

- Requirement 1:
  Lightweight for local analysis
- Requirement 2:
  Accurate enough for various logs
- Requirement 3:
  Adaptive to unseen logs

ML method

+ Lightweight
- Needs extensive training data
- Not adaptive

Filter massive normal log messages

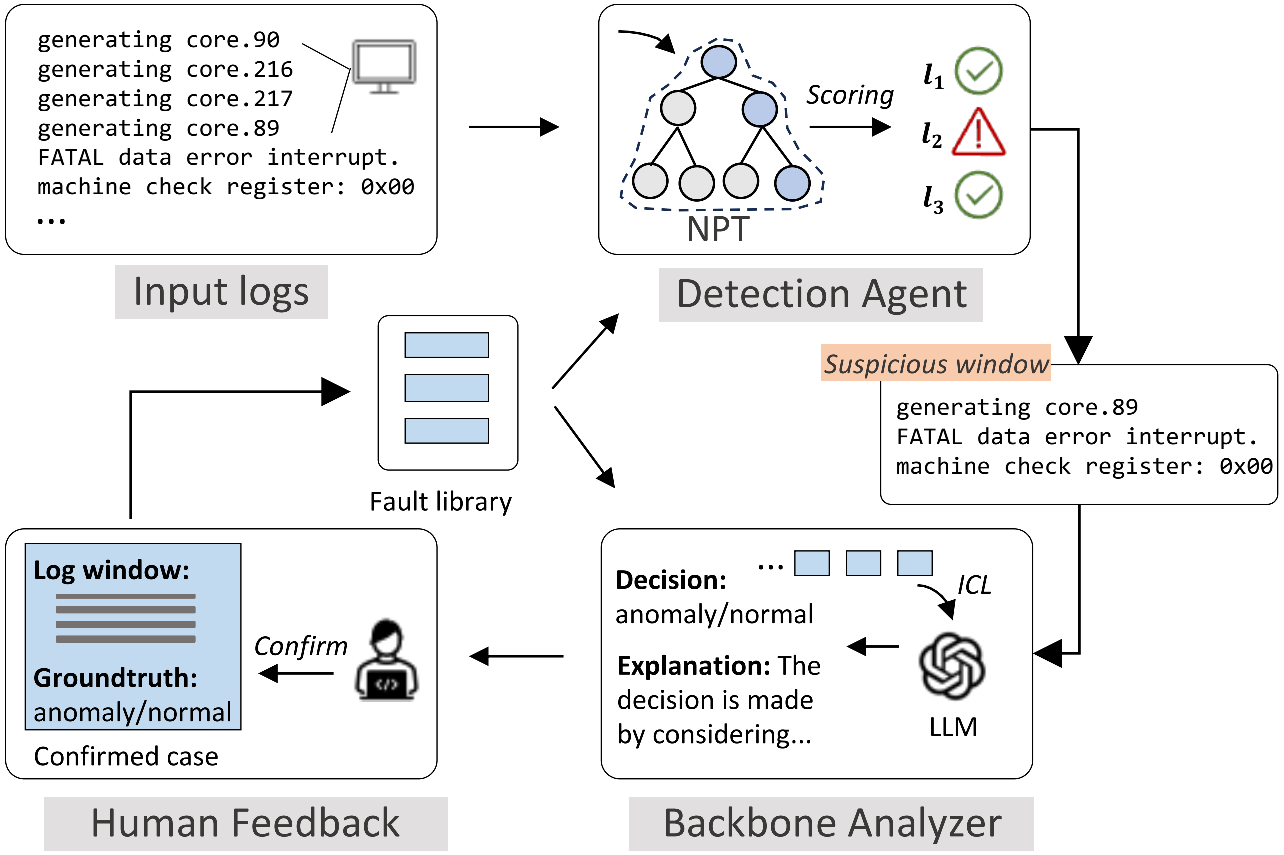
Large language models

+ Semantic comprehension
+ Zero/few-shot prediction
+ Follow instructions
- Slow
- High cost

Analyze only suspicious log messages in detail

Our synergistic approach: Sealog

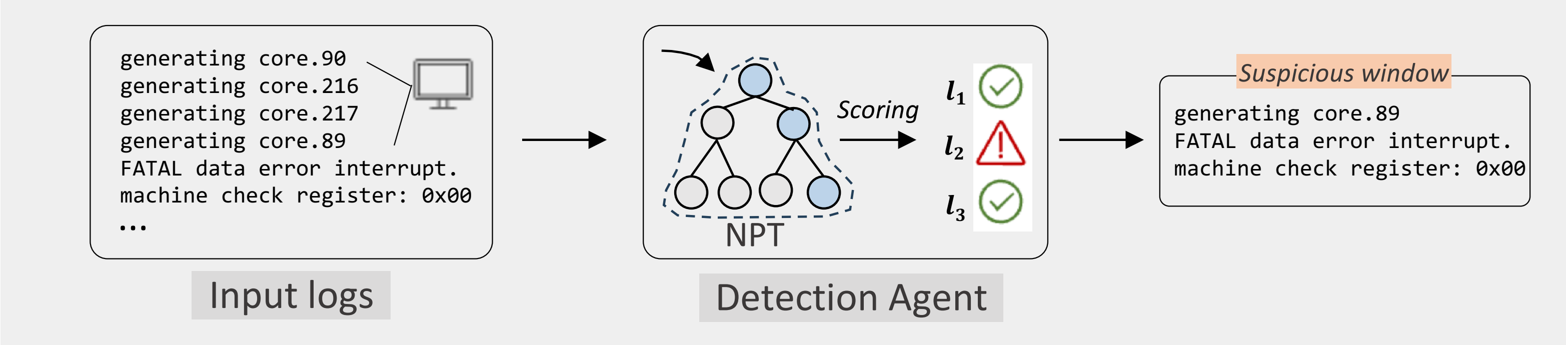
- Integrating large language models (LLM) with lightweight ML methods



Overall Framework of Sealog

Detection agent of Sealog

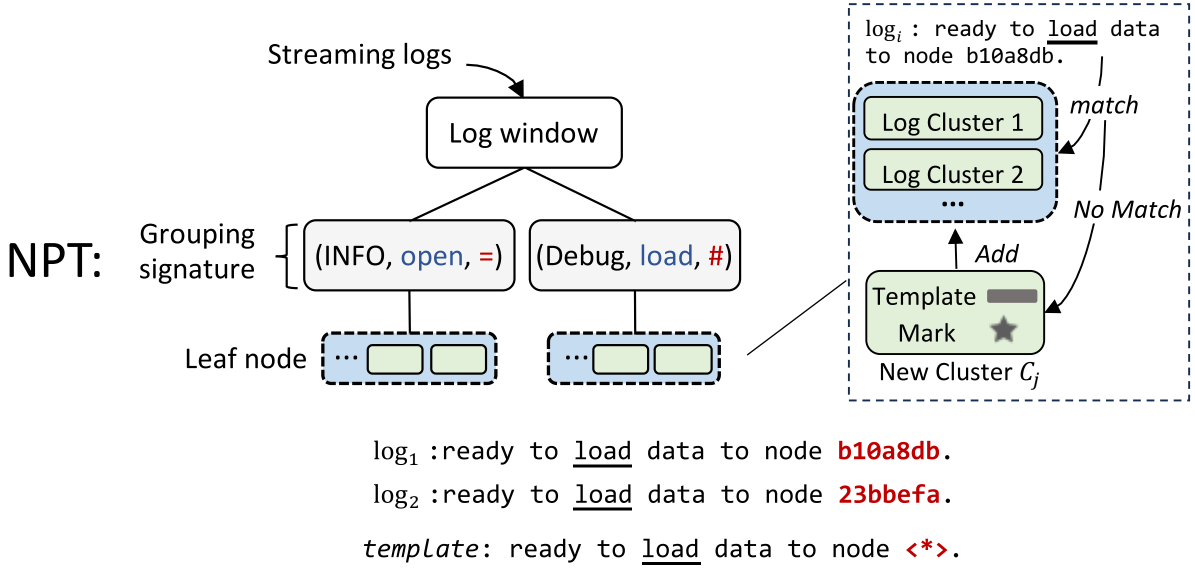
- Detection agent (N-gram probabilistic tree, NPT)

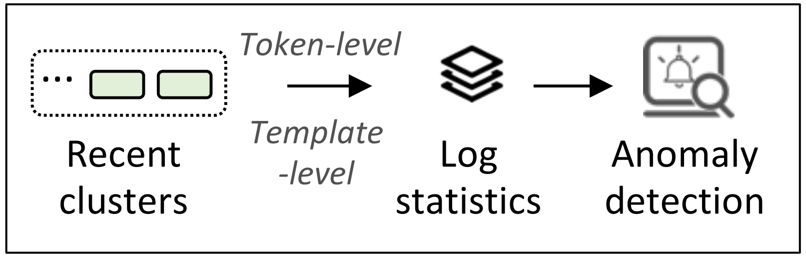


① Parse logs to event templates

② Deploy locally

③ Ensure efficiency and high recall





Logs with unseen tokens: $P(c\,|\,X) = \dfrac{\#\ unseen\ tokens}{\#\ Total\ tokens}$

Otherwise: naive bayes classifier

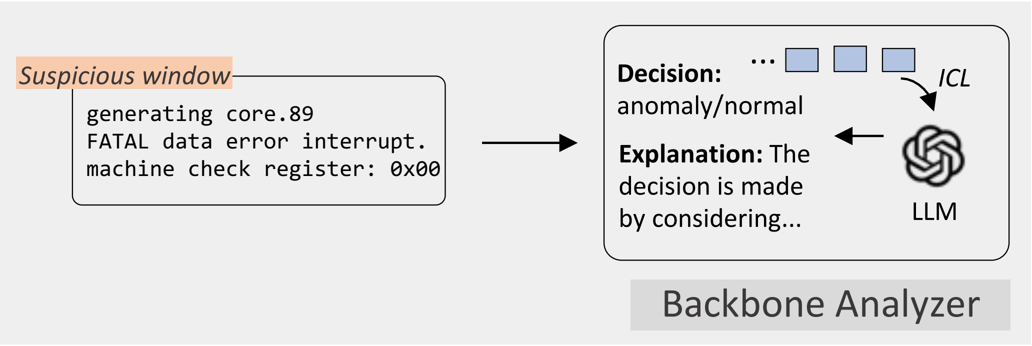
$$P(c\,|\,X) = P(x_1\,|\,c) \times P(x_2\,|\,c) \times \cdots \times P(x_n\,|\,c) \times P(c)$$

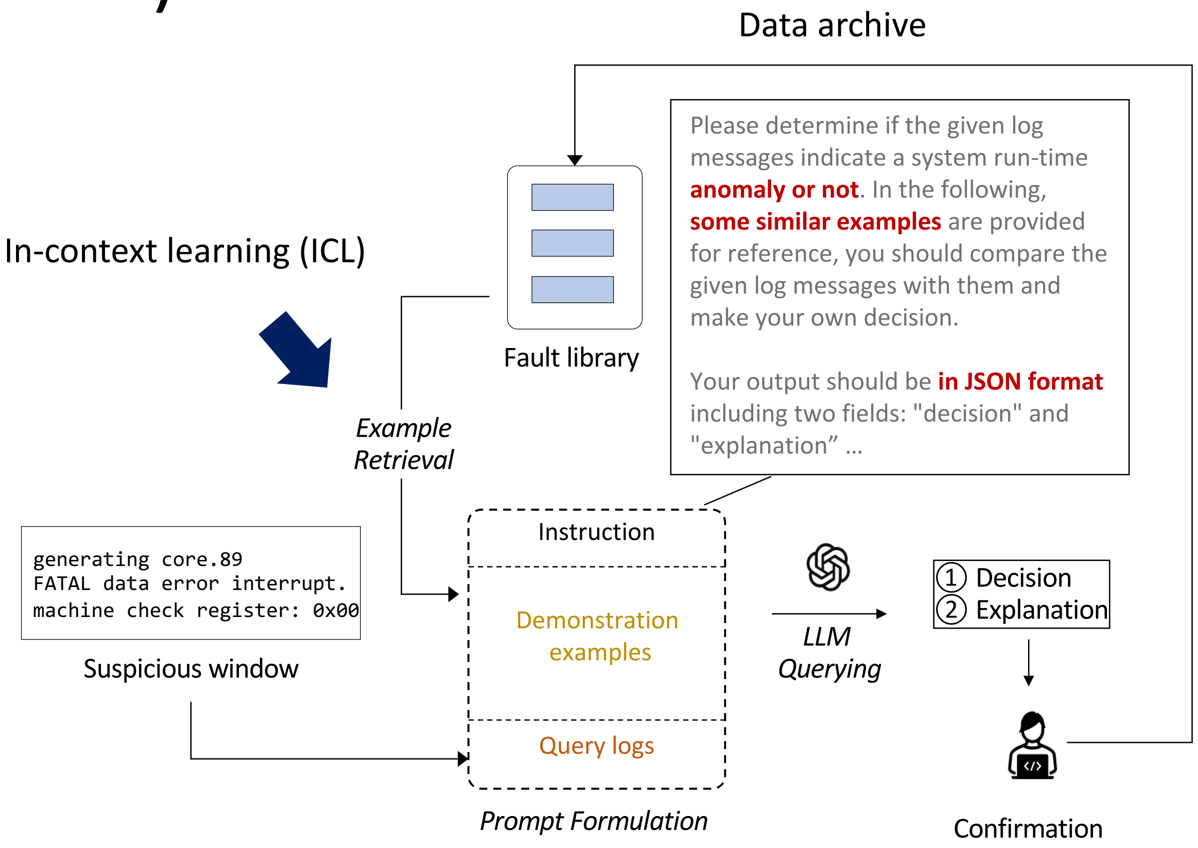
Anomaly / Normal

Log n-gram tokens

Backbone analyzer of Sealog

- Backbone Analyzer (ICL-enhanced LLM)





① Understand log semantics

② Deployed as a remote service

③ Receive limited queries from detection agent

Evaluation

- Datasets

Dataset	BGL	Thunderbird	Industry
# Log messages	4,747,963	10,000,000	1,488,073
# Templates	456	1,504	3,241
# Train windows (anomaly ratio)	2,884 (21%)	416 (55%)	3,048 (13%)
# Test windows (anomaly ratio)	722 (24%)	105 (30%)	933 (18%)

Industry

- Real-world data from Huawei Cloud
- 103 types of anomalies
- Labeled by on-site engineers

- Research Questions
  - RQ1: How effective is SeaLog under the offline setting?
  - RQ2: How effective is SeaLog under the online setting?
  - RQ3: How does the number of queries affect the performance of SeaLog?
  - RQ4: How efficient is Sealog?

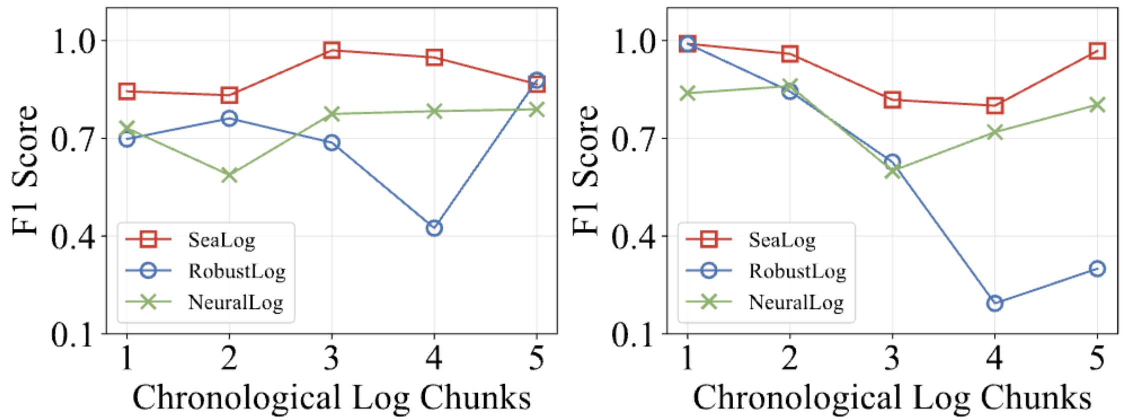
- Metrics
  - Precision: $\frac{TP}{TP + FP}$
  - Recall: $\frac{TP}{TP + FN}$
  - F1 Score: $\frac{2 \times precision \times recall}{precision + recall}$

Evaluation

- RQ1: Offline Effectiveness

Method	BGL			Thunderbird			Industry		
	Precision	Recall	**F1 score**	Precision	Recall	**F1 score**	Precision	Recall	**F1 score**
IF	0.125	0.615	0.208	0.291	0.968	0.448	0.176	0.994	0.299
LR	0.738	0.437	0.549	0.842	0.516	0.640	0.818	0.655	0.727
DT	1.000	0.570	0.726	1.000	0.839	0.912	0.942	0.788	0.858
DeepLog	0.241	0.895	0.380	0.295	1.000	0.456	0.358	0.909	0.513
LogAnomaly	0.268	0.862	0.409	0.307	1.000	0.470	0.360	0.927	0.519
RobustLog	0.942	0.961	0.951	1.000	0.710	0.830	0.984	0.764	0.860
NeuralLog	0.881	0.886	0.883	0.713	0.719	0.715	0.889	0.895	0.887
SeaLog	0.994	0.991	**0.993**	1.000	0.903	**0.949**	1.000	0.931	**0.964**

- RQ2: Online Effectiveness



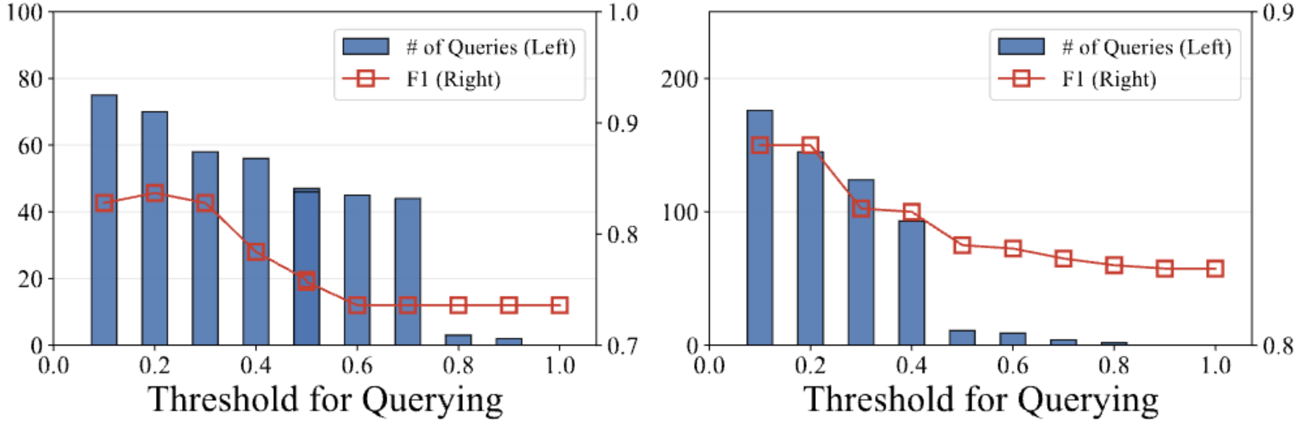
(a) BGL (b) Industry

Observation 1: Sealog is the most effective solution in the offline setting.

Observation 2: Sealog keeps a high performance in the online setting.

Evaluation

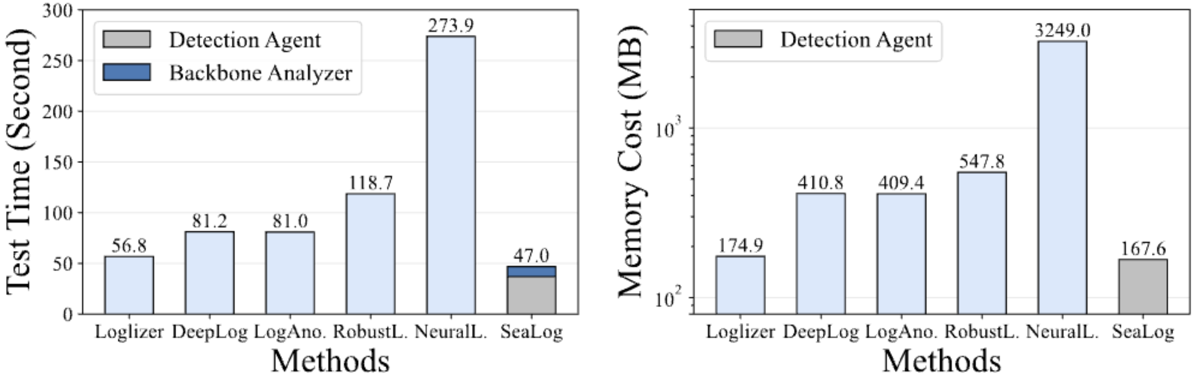
- RQ3: Impact of query numbers



(a) BGL (b) Industry

Observation 3: Only limited queries are forwarded to LLM.

- RQ4: Time and Memory Efficiency

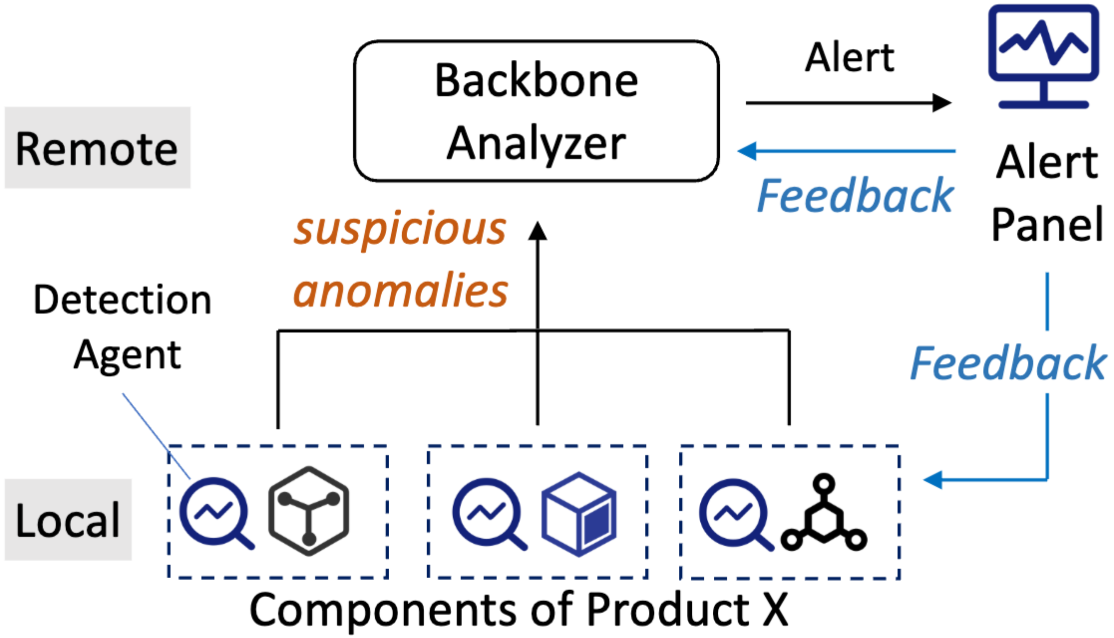


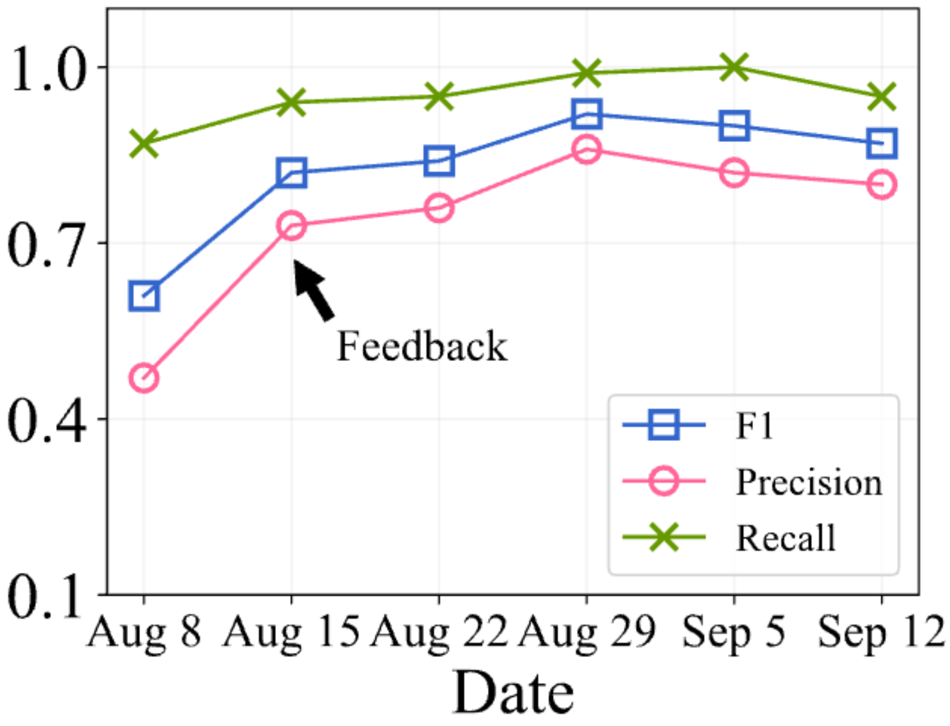
(a) Time Efficiency Comparison (b) Space Efficiency Comparison

Observation 4: SeaLog demonstrates high efficiency in both time and memory consumption.

Industry deployment

- Deployment in Huawei Cloud





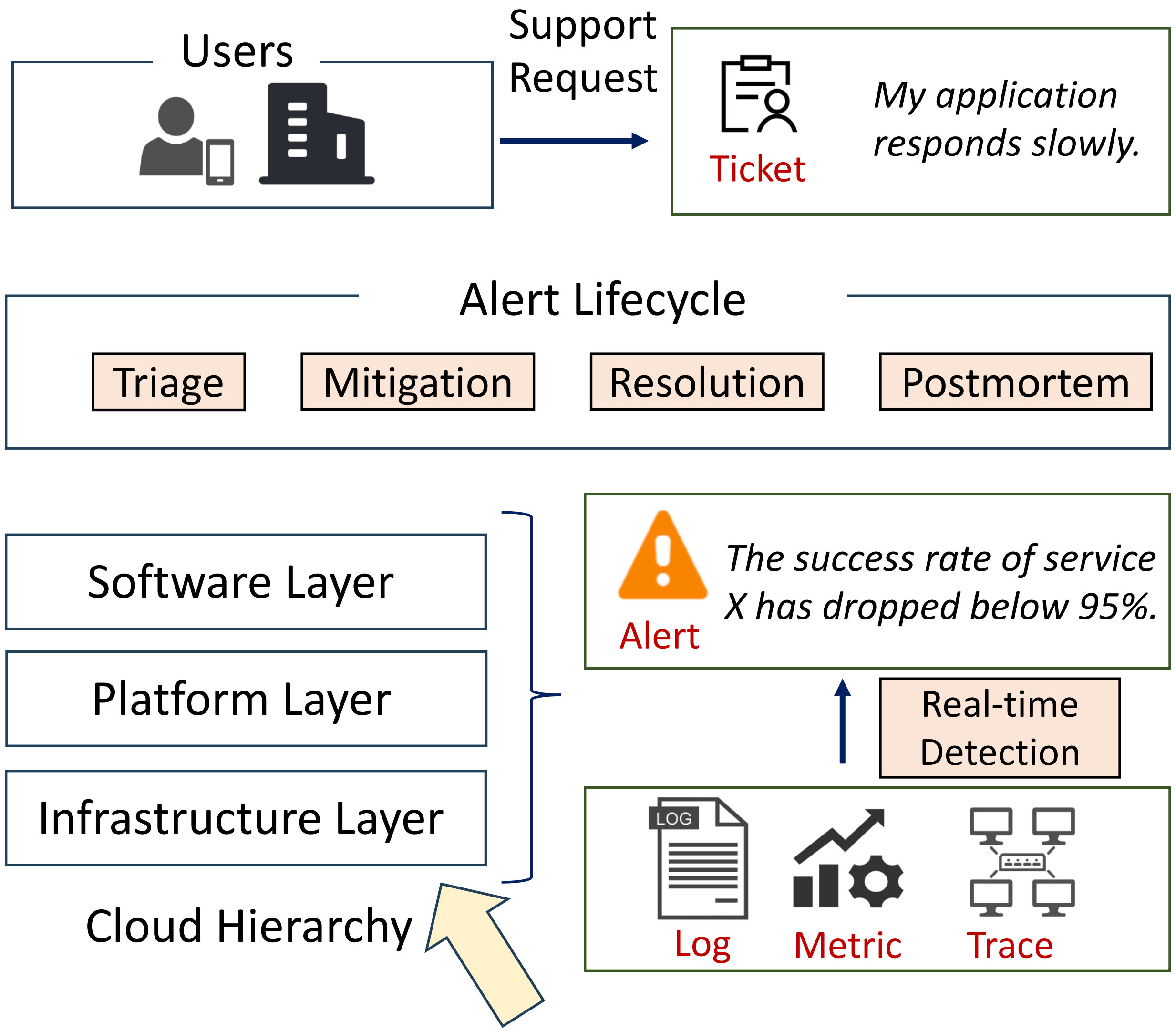
Summary of ① Sealog

- Log-based anomaly is essential, which requires an anomaly detector accurate, lightweight and adaptive.

- We propose Sealog, a synergistic approach integrating both the advantages of ML-based and LLM-based methods.

- Sealog fulfills these three requirements and has been deployed in real-world production environment.

Our goal: Intelligent reliability management



Thesis Contribution

① Sealog

Scalable and adaptive log-based anomaly detection

② Prism

Improving observability across different layers of cloud hierarchy

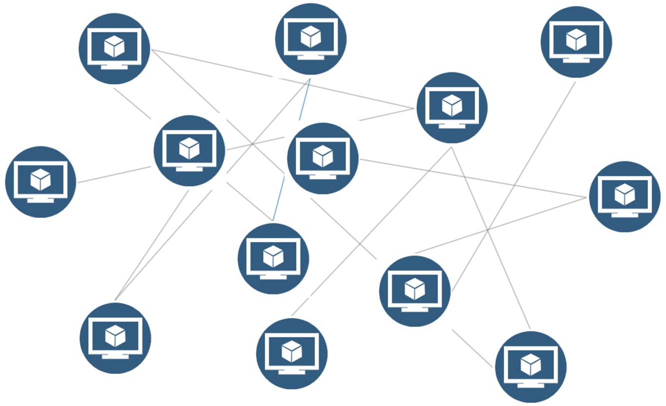
③ iPACK

Correlating tickets and alerts for more comprehensive ticket deduplication

Black-box view of cloud vendors facing millions of instances

Applications

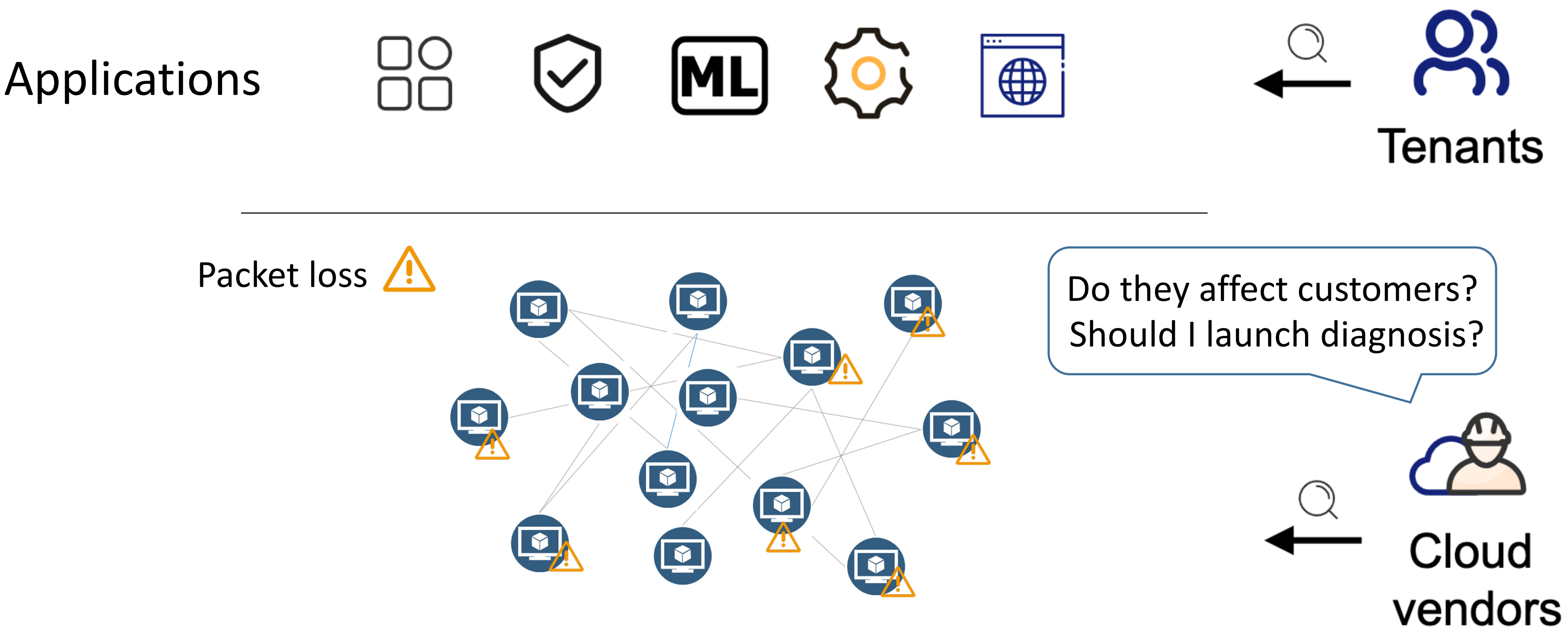
Tenants



Cloud vendors

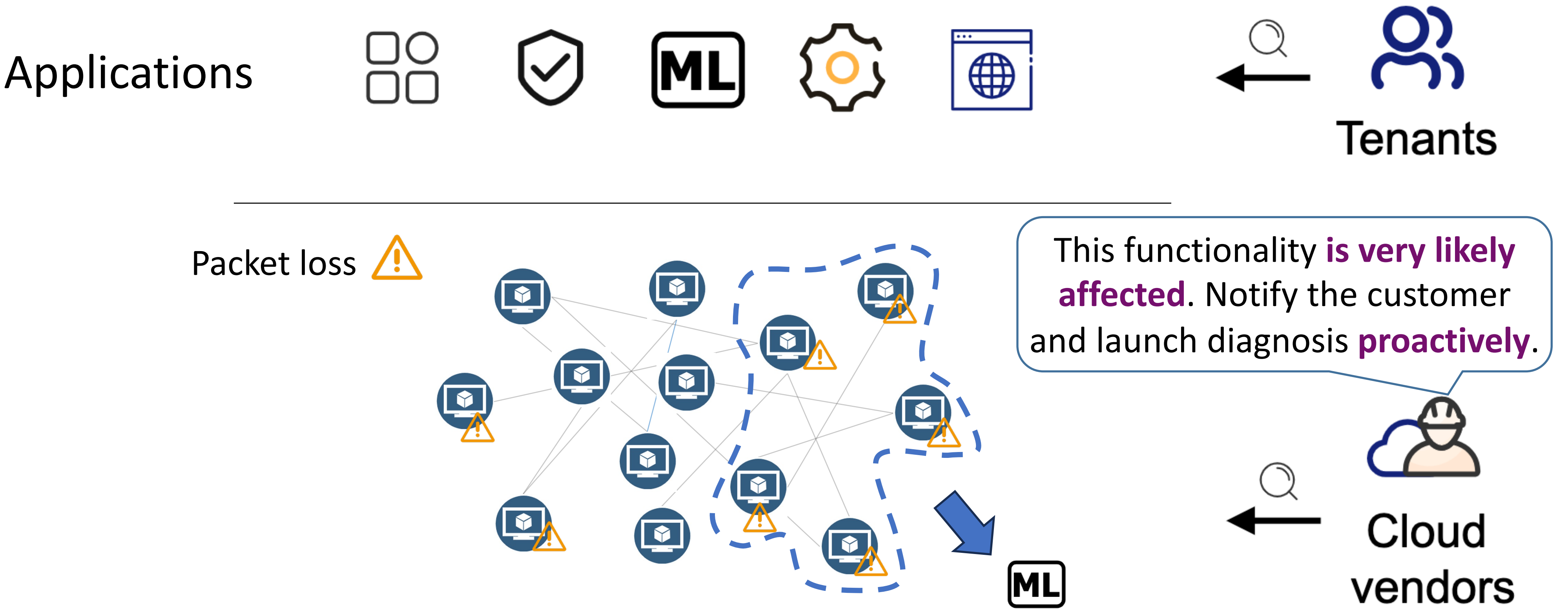
Massive **Black-box** Instances
(typically millions of)

A Motivating Example



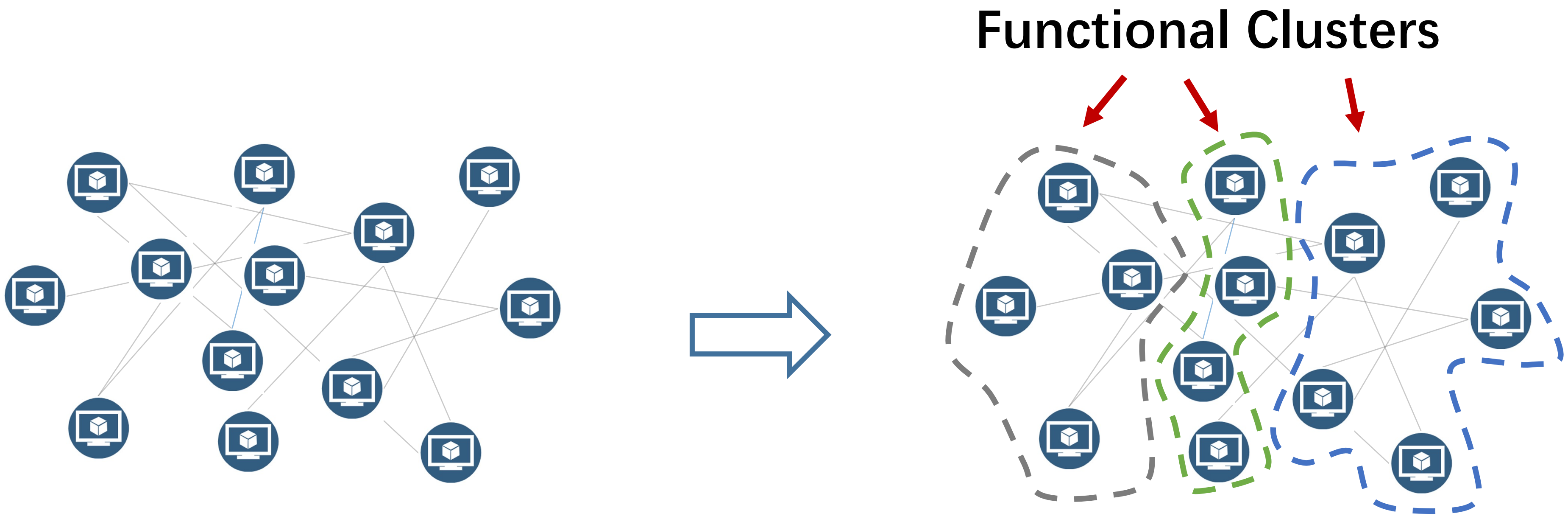
Massive **Black-box** Instances
(typically millions of)

A Motivating Example



Clustered Instances
(Serving the same functionalities)

Our Problem



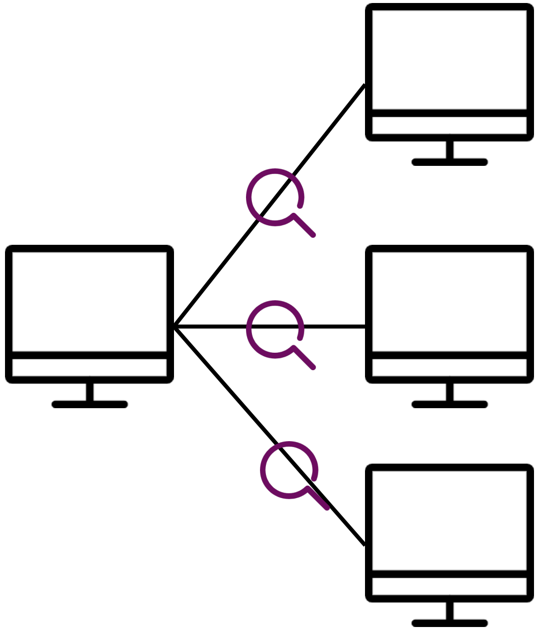
Massive **Black-box** Instances
(typically millions of)

Clustered Instances
(Serving the same functionalities)

Problem: How do we find **functional clusters** in massive instances **with ONLY data visible to cloud vendors** (with customers' consent)?

Data visible to cloud vendors

- Two types of typical monitoring data



Trace: (srcIp, dstIp, srcPort, dstPort)

Communication Traces

cpu utilization
packet received
packet sent
mem usage
disk read
disk write
…

Monitoring Metrics

A Pilot Study

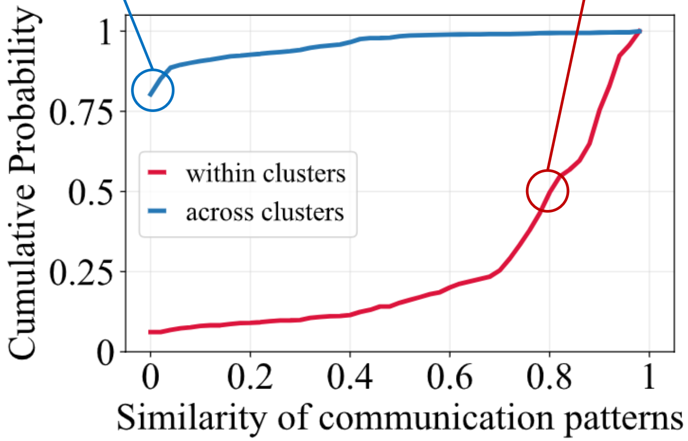
- 3,062 internal instances covering 397 types of functionalities

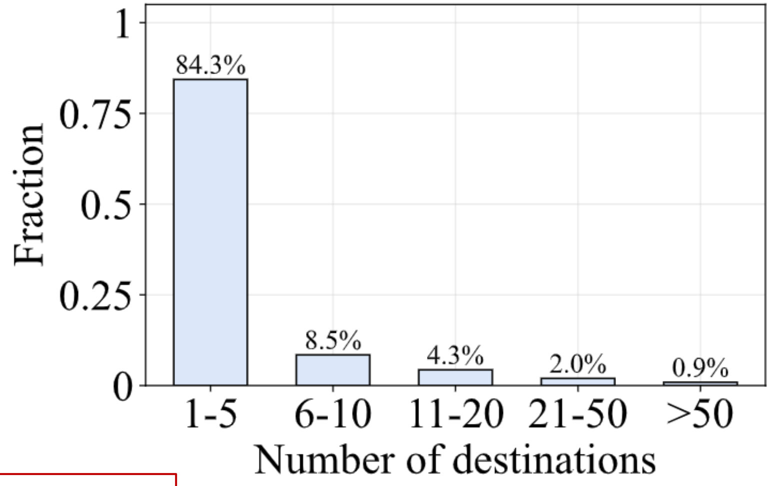
Huawei Cloud

> 75% across-cluster instances have nearly zero similarity.

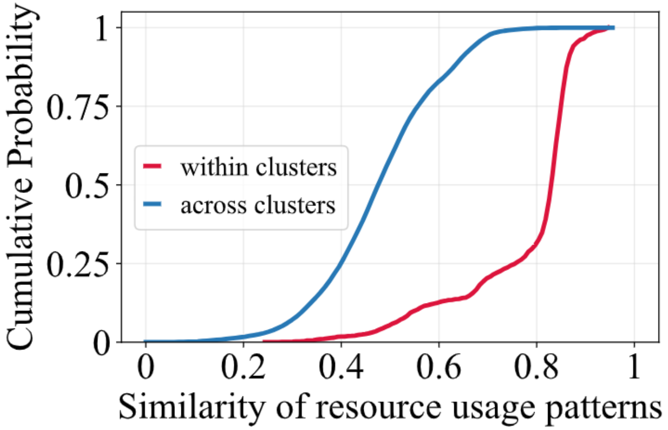
> 50% in-cluster instances have > 0.8 similarity.

Communication Similarity





Resource Utilization Similarity



Findings

- In-cluster instances share **similar communication and resource patterns**.
- Most instances only communicate with a small number of instances (**locality**).
- Both data are **noisy**.

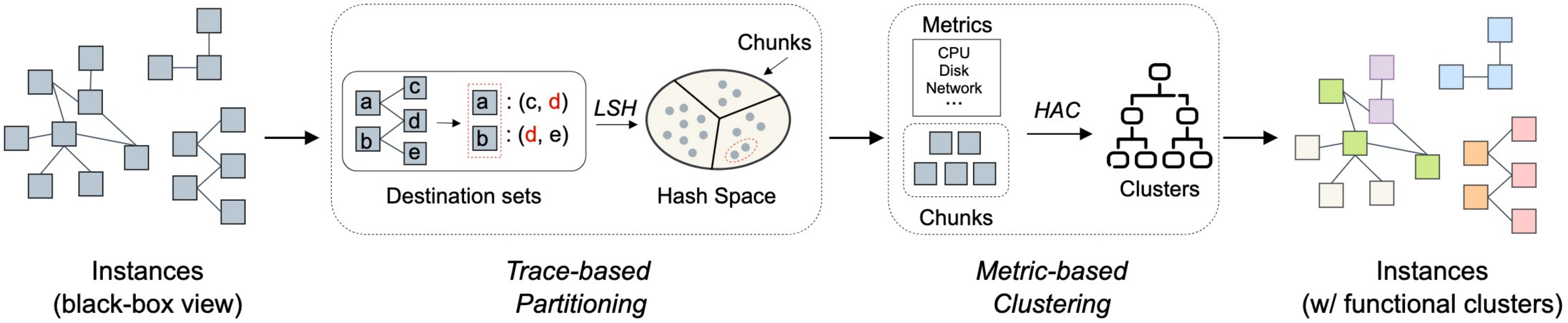
Method

Problem: How do we find **functional clusters** in massive instances **with ONLY data visible to cloud vendors** (with customers' consent)?

Challenges:

- Massive instances (typically millions in cloud systems)
- Limited noisy monitoring data for cloud vendors

Our Solution: **Prism**



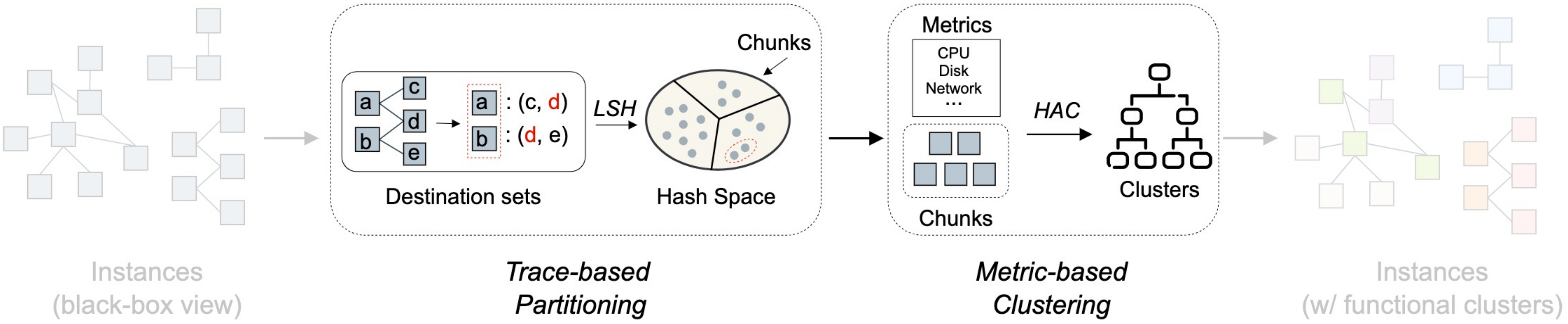
Method

Problem: How do we find **functional clusters** in massive instances **with ONLY data visible to cloud vendors** (with customers' consent)?

Challenges:

- Massive instances (typically millions in cloud systems)
- Limited noisy monitoring data for cloud vendors

Our Solution: **Prism**

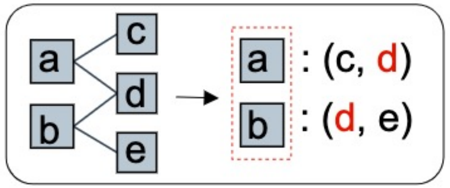


Method

Trace-based Partitioning

Input:
- All instances
- Communication traces

Output:
- Coarse-grained chunks



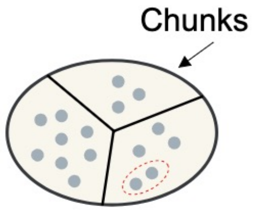
Destination Sets

Jaccard similarity:

$$J(x_i, x_j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|}$$

Pairwise comparison

Efficient Locality Sensitive Hashing



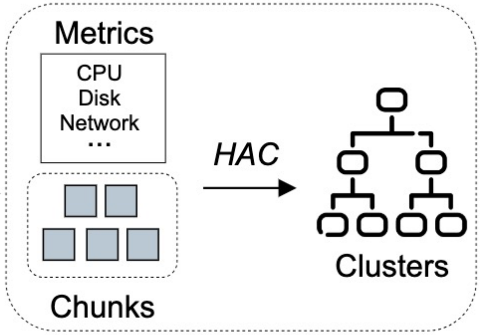
Hash Space

Strong Locality!

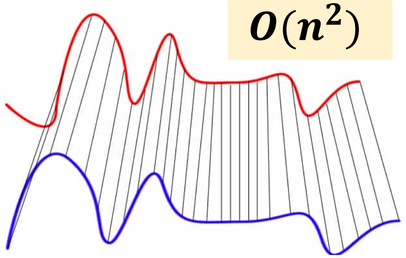
Metric-based Clustering

Input:
- Coarse-grained chunks
- Monitoring metrics (cpu, mem, disk, etc.)

Output:
- Functional clusters



$\boldsymbol{O(n^2)}$



Dynamic Time Warping (DTW) Distance

Apply independently for each small chunk (<=50 instances)

Evaluation

- Datasets

Datasets	# Functionalities	# Instances	# Traces	# Metrics
Dataset $\mathcal{A}$	292	2,035	100.2 M	7.25 M
Dataset $\mathcal{B}$	105	1,027	121.6 M	3.71 M
Total	397	3,062	212.6 M	10.96 M

- Real-world data from Huawei Cloud
- Manually labeled internal instances

- Research Questions
  - RQ1: What is the **effectiveness** of Prism?
  - RQ2: What is the **contribution of each component**?
  - RQ3: What is the **impact of parameter settings**?
  - RQ4: What is the **efficiency** of Prism?

- Metrics
  - Homogeneity: how precise?
  - Completeness: how complete?
  - V-measure: a balanced metric

Evaluation

- RQ1: Effectiveness

Methods	Dataset $\mathcal{A}$ Homo.	Dataset $\mathcal{A}$ Comp.	Dataset $\mathcal{A}$ V Meas.	Dataset $\mathcal{B}$ Homo.	Dataset $\mathcal{B}$ Comp.	Dataset $\mathcal{B}$ V Meas.
OSImage	0.238	0.894	0.376	0.258	0.889	0.400
CloudCluster	0.346	0.748	0.473	0.369	0.753	0.495
ROCKA	0.831	0.882	0.856	0.875	0.900	0.887
OmniCluster	0.932	0.862	<u>0.896</u>	0.944	0.877	<u>0.909</u>
Prism	0.976	0.916	**0.945**	0.979	0.922	**0.950**

Observation 1: Prism outperforms all state-of-the-art comparative methods.

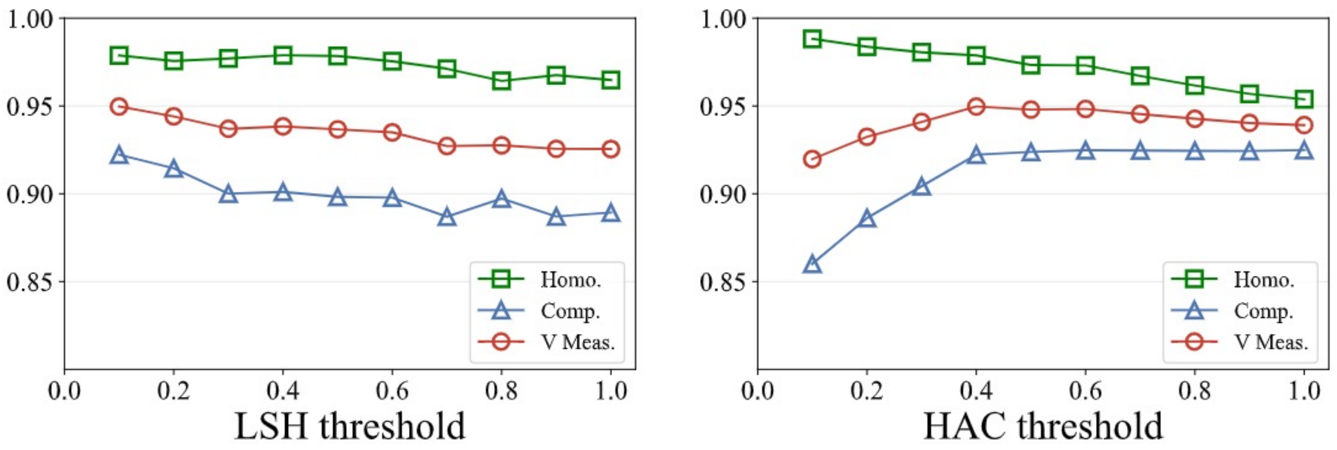
- RQ2: Ablation

Methods	Dataset $\mathcal{A}$ Homo.	Dataset $\mathcal{A}$ Comp.	Dataset $\mathcal{A}$ V Meas.	Dataset $\mathcal{B}$ Homo.	Dataset $\mathcal{B}$ Comp.	Dataset $\mathcal{B}$ V Meas.
Prism	0.976	0.916	**0.945**	0.979	0.922	**0.950**
Prism w/o Metrics	0.462	**0.920**	0.615	0.463	**0.949**	0.622
Prism w/o Traces	0.949	0.869	<u>0.907</u>	0.915	0.893	<u>0.904</u>

Observation 2: Both components contribute to the overall performance.

Evaluation

- RQ3: Parameter Sensitivity



Observation 3: Prism is robust to threshold settings for both LSH and HAC.

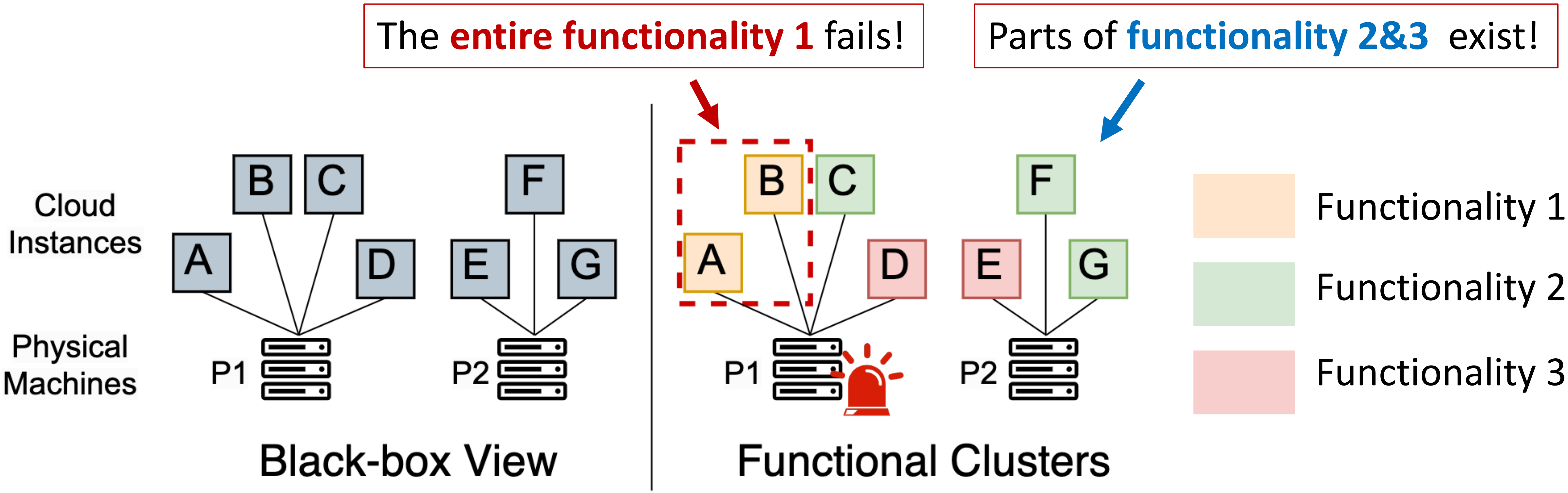
- RQ4: Efficiency

Methods	# Instances 1,000	5,000	10,000	50,000	100,000
CloudCluster	0.9	23.87	78.65	1768.7	5585.7
ROCKA	80.7	1981.8	7850.3	-	-
OmniCluster	31.7	264.6	1048.6	26531.8	-
Prism w/o Metrics	3.9	19.1	40.2	195.1	392.4
Prism w/o Traces	80.3	2066.1	8232.3	-	-
Prism	18.2	89.4	183.9	929.2	1912.7

Observation 4: Prism can efficiently handle massive instances in cloud systems.

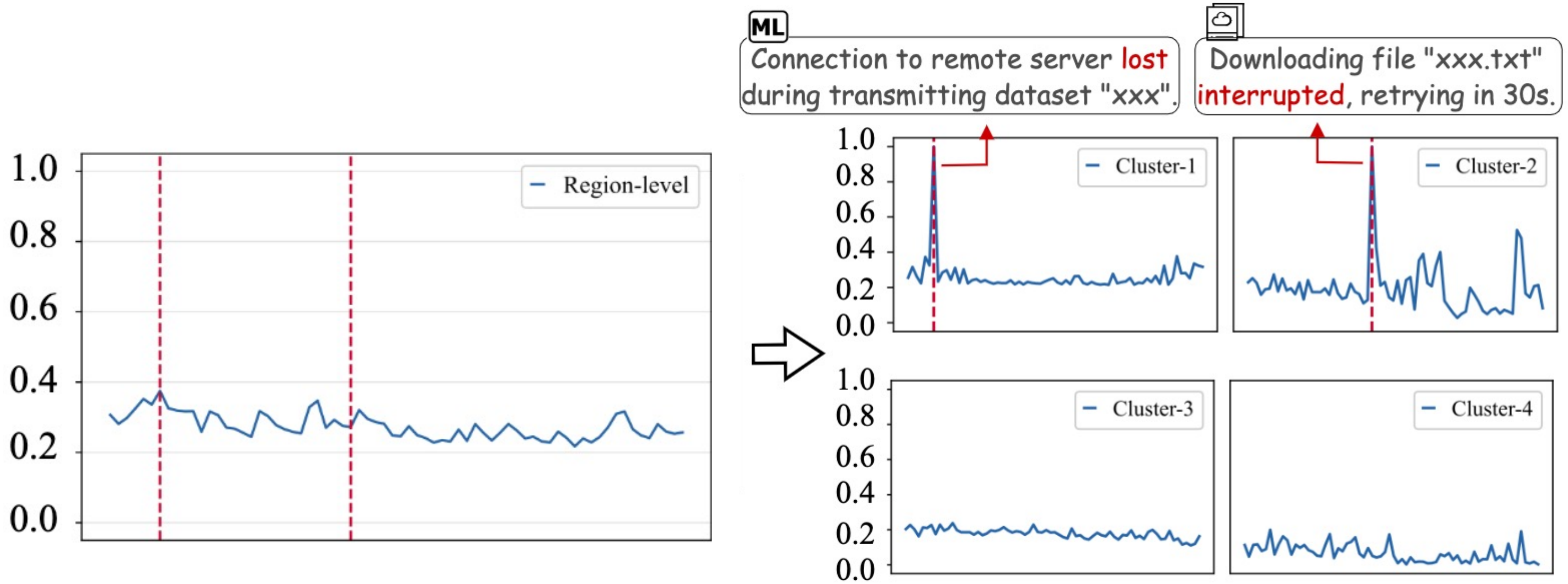
Industrial Experience

- Use case 1: vulnerable deployment identification



Industrial Experience

- Use case 2: latent issue discovery



Aggregated by **region**

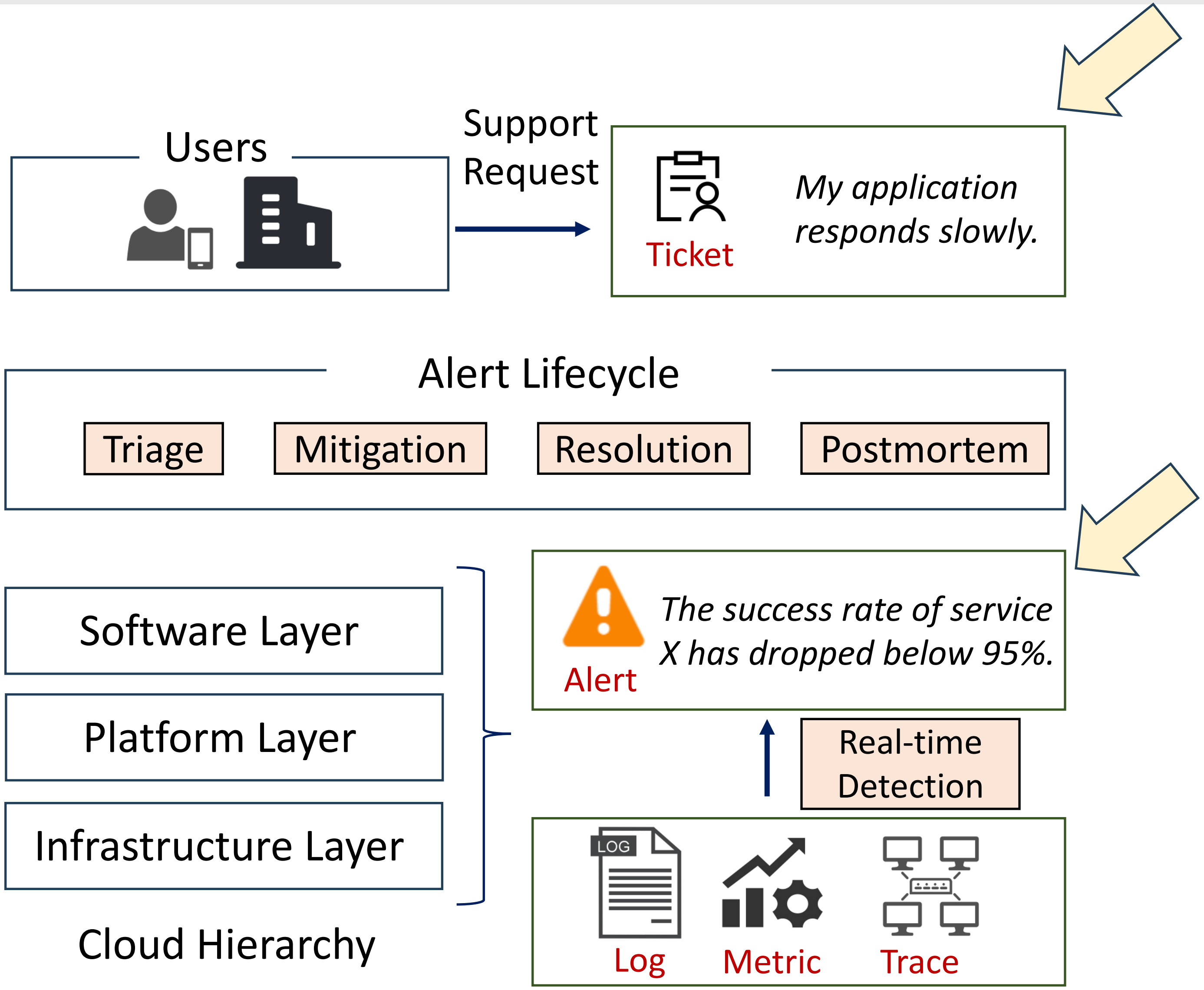
Aggregated by **functional clusters (Prism applied)**

Fragmented lost packets of massive instances

Summary of ② Prism

- Virtualization technologies improve resource utilization but lead to limited observability in the cloud.
- The proposed **Prism** reveals functional clusters by leveraging communication patterns and resource patterns among instances.
- Prism is effective and efficienct, which provides additional insights for enhanced cloud reliabiilty.

Our goal: Intelligent reliability management



Thesis Contribution

① Sealog

Scalable and adaptive log-based anomaly detection

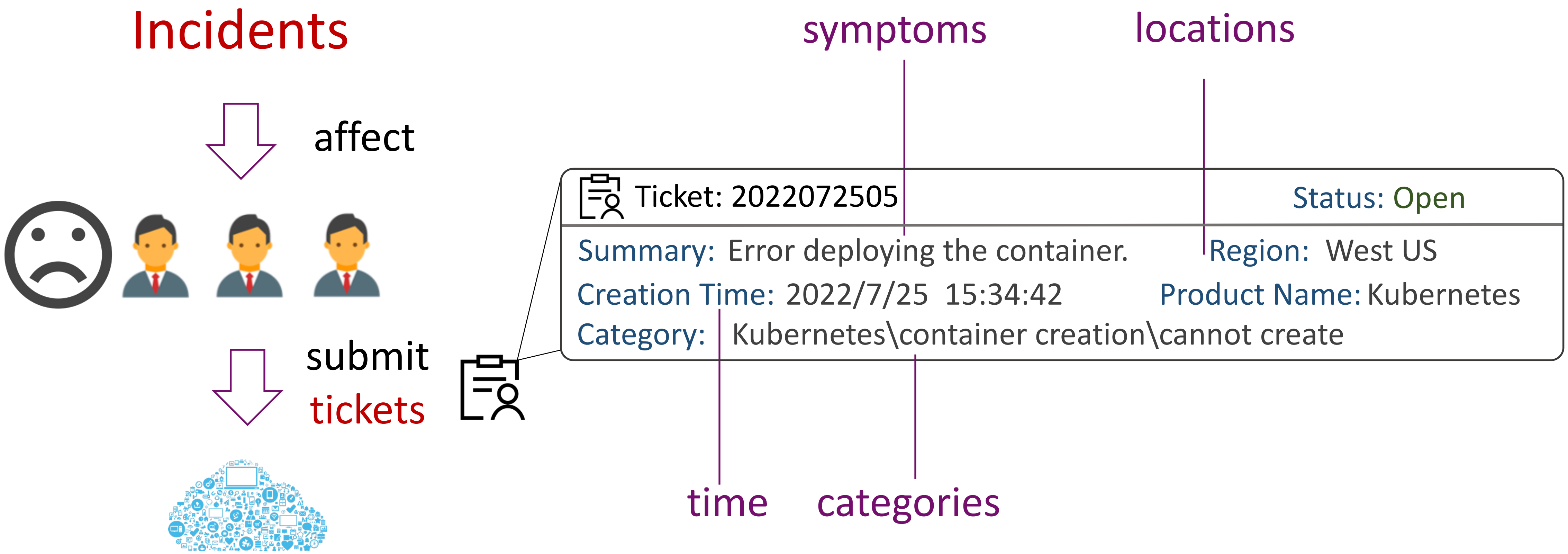
② Prism

Improving observability across different layers of cloud hierarchy

③ iPACK

Correlating tickets and alerts for more comprehensive ticket deduplication

Customers submit tickets to cloud vendors for help



Customers submit tickets to cloud vendors for help

Incidents

affect

submit
tickets

Thousands of services

Hundreds of millions of customers

Overwhelming tickets

Customers submit tickets to cloud vendors for help

Incidents are Inevitable!

affect

submit tickets

Thousands of services

Hundreds of millions of customers

Overwhelming tickets

Problem: how to identify duplicate tickets?

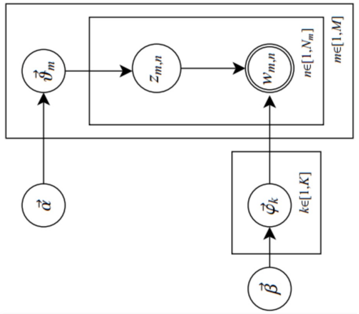
Limitations of text similarity-based solutions

- Intuition: duplicate tickets have similar semantics

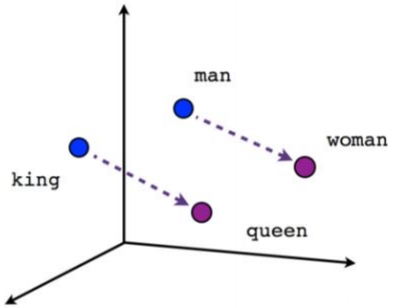
The app service stopped working.

Failed accessing web app.

The web app is unavailable.



Topic modeling-baseds methods
(ICSE’18)



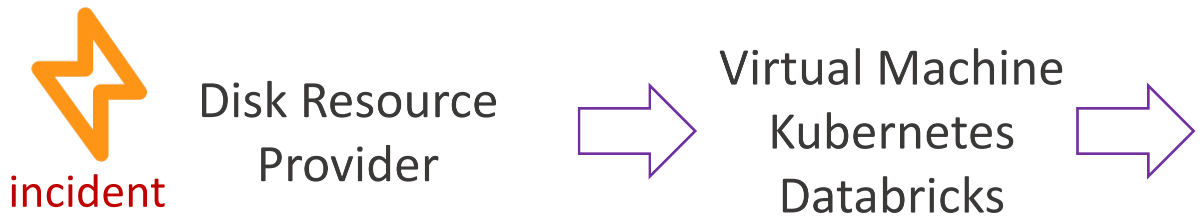
Word2Vec-based methods
(ASE’19)

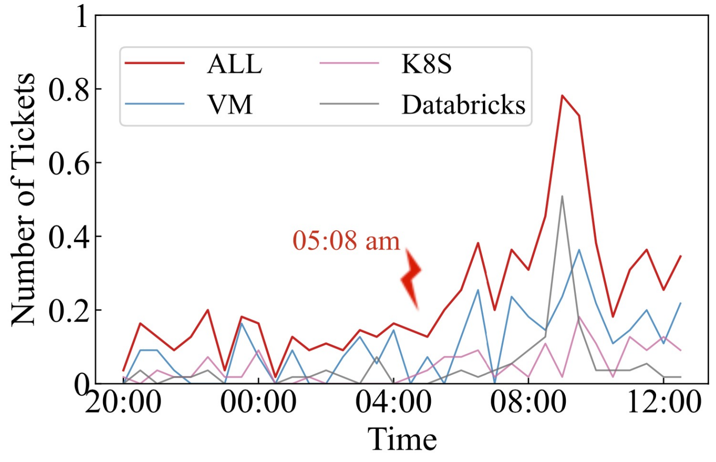
BERT-based methods
(ICSE’21)

[ICSE18] Lwe: Lda refined word embeddings for duplicate bug report detection
[ASE19] iFeedback: exploiting user feedback for real-time issue detection in large-scale online service systems.
[ICSE21] Automatically matching bug reports with related app reviews

Limitations of text similarity-based solutions

- However, duplicate tickets in cloud may have distinct semantics



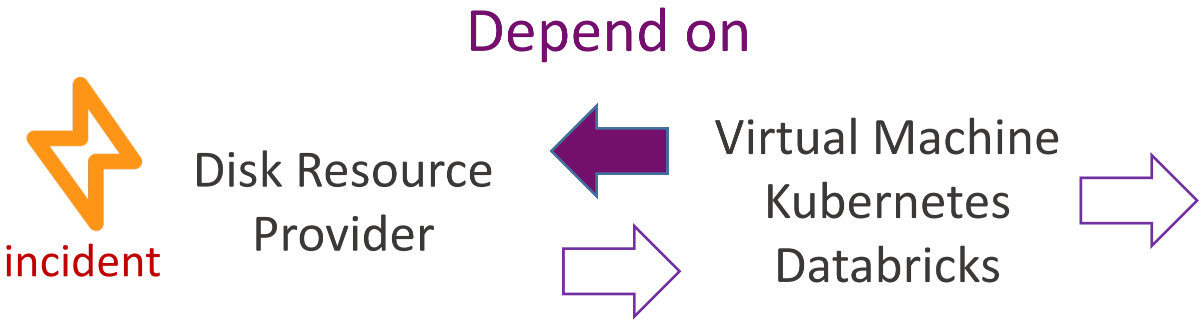


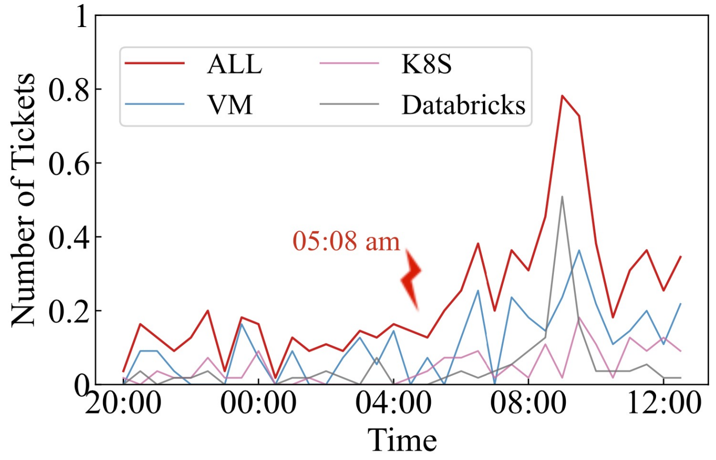
Service	**Tickets**	
	Category	Summary
VM	VM/Scale Update	t_1: Virtual machine scale sets resize issue.
	VM/VM Start	t_2: Server did not start on time.
Databricks	Databricks/Job Issue	t_3: Unable to open cluster of Databricks.
	Databricks/Cluster Launch	t_4: Unable to provision clusters.
K8S	K8S/Cluster Update	t_5: Unable to autoscale.
	K8S/Cluster Update	t_6: Cannot upgrade node pool, stuck.

diverse symptoms!

Limitations of text similarity-based solutions

- Reason: dependency between different services





Service	Tickets	
	Category	Summary
VM	VM/Scale Update	t_1: Virtual machine scale sets resize issue.
	VM/VM Start	t_2: Server did not start on time.
Databricks	Databricks/Job Issue	t_3: Unable to open cluster of Databricks.
	Databricks/Cluster Launch	t_4: Unable to provision clusters.
K8S	K8S/Cluster Update	t_5: Unable to autoscale.
	K8S/Cluster Update	t_6: Cannot upgrade node pool, stuck.

diverse symptoms!

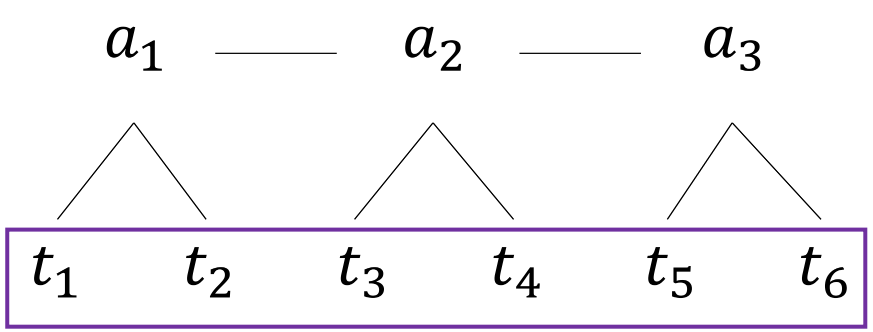
Our solution: two-stage linking

- Leverage cloud runtime information: Alerts

Service	Tickets		Alerts	
	Category	Summary	Component	Title
VM	VM/Scale Update	t_1: Virtual machine scale sets resize issue.	Resource Provider	a_1 : VMStart Failures exceed 300 times.
	VM/VM Start	t_2: Server did not start on time.		
Databricks	Databricks/Job Issue	t_3: Unable to open cluster of Databricks.	Control Plane	a_2 : Databricks cluster creation fails.
	Databricks/Cluster Launch	t_4: Unable to provision clusters.		
K8S	K8S/Cluster Update	t_5: Unable to autoscale.	Resource Scheduler	a_3 : The PUT operation success rate $<80\%$.
	K8S/Cluster Update	t_6: Cannot upgrade node pool, stuck.		a_4 : CPU utilization exceeds 90%.

① alert — alert

② alert | tickets



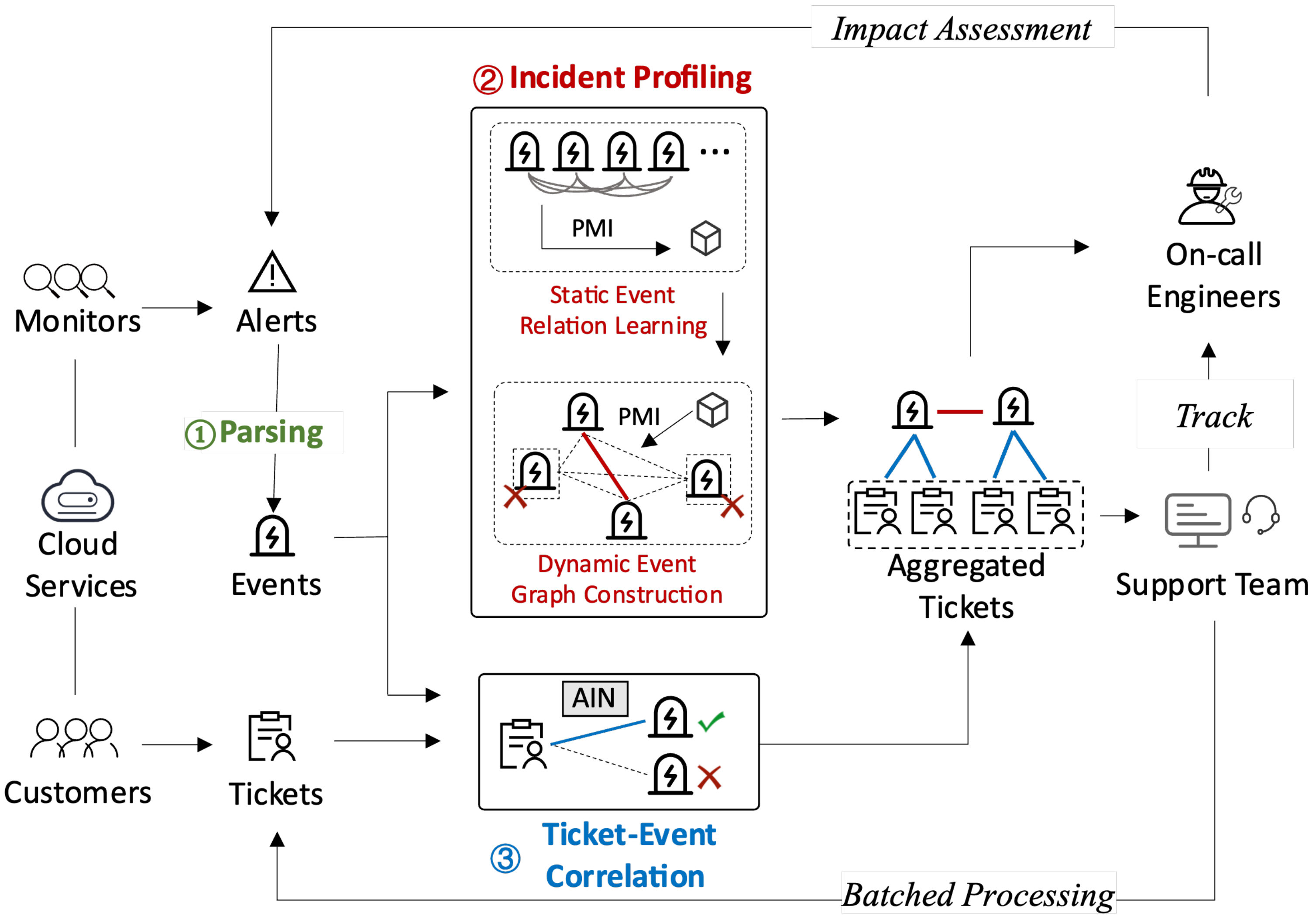
Challenge 1 Alerts are massive and noisy

- Indicative alerts: a_1 a_2 a_3
- Regular alerts: a_4

Challenge 2 High feature cardinality Large Combinations!

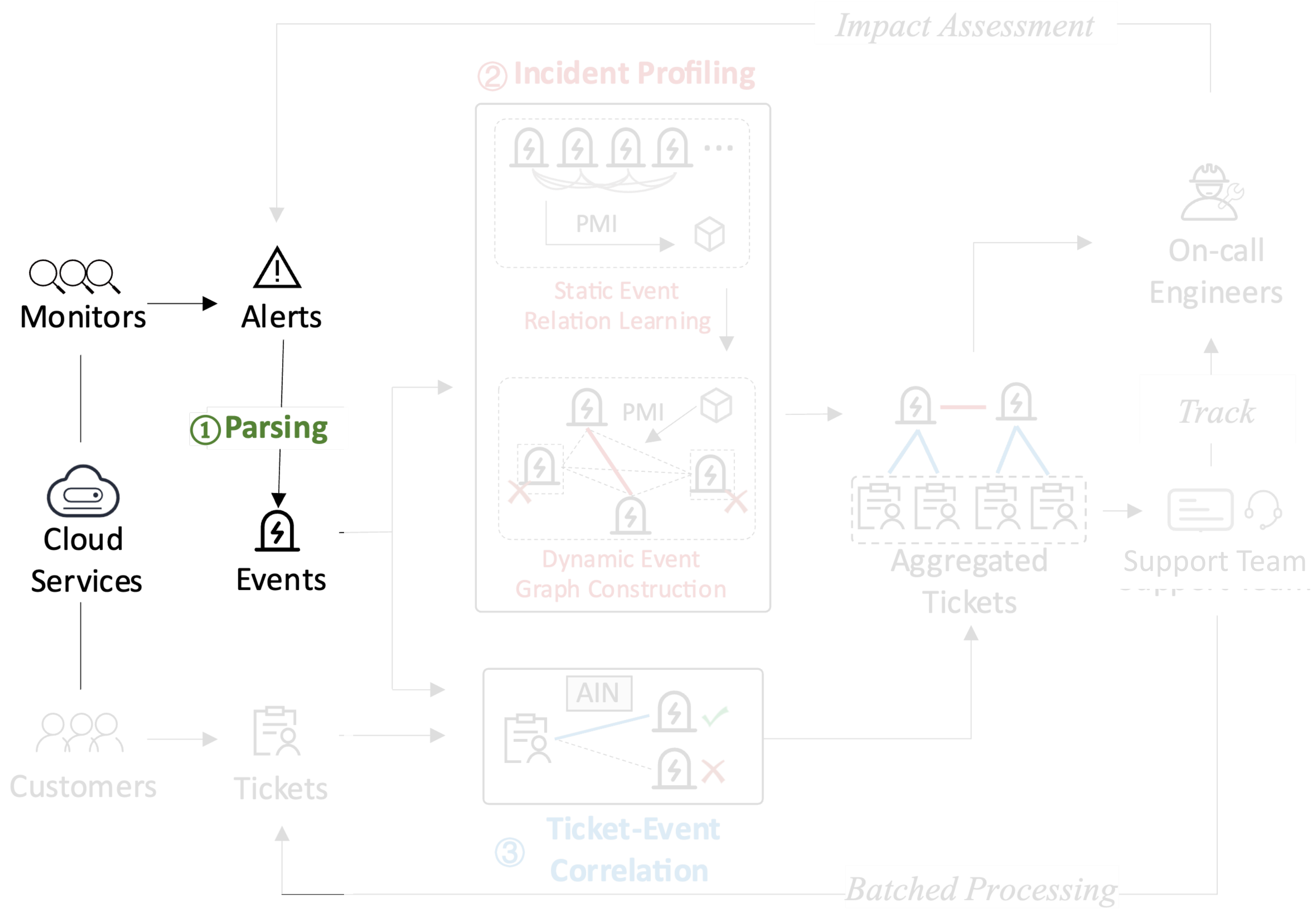
- Alerts : free text, 2000+ components, 10000+ IDs, etc
- Tickets: free text, 3000+ categories, etc

Overall Framework of iPACK



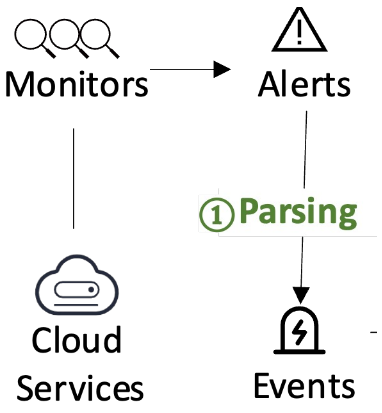
Overall Framework of iPACK

① Reduce redundant alerts



Overall Framework of iPACK

① Reduce redundant alerts



⚠ VMStart Failures exceed 100 times
⚠ VMStart Failures exceed 150 times
⚠ VMStart Failures exceed 200 times
⚠ VMStart Failures exceed 250 times

Drain[1]
(log parser)

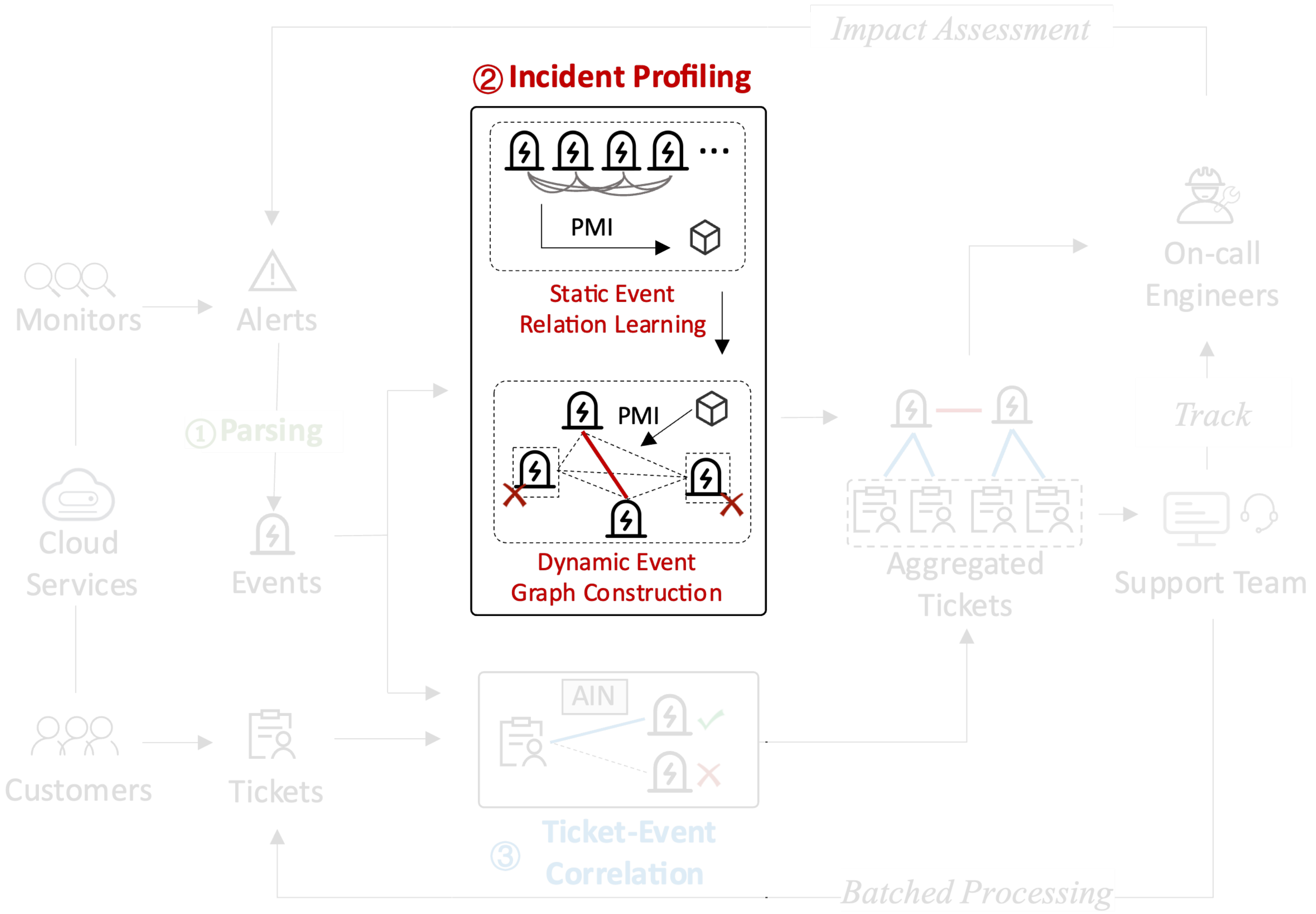
Events VMStart Failures exceed <*> times

[1] Drain: An Online Log Parsing Approach with Fixed Depth Tree

Overall Framework of iPACK

① Reduce redundant alerts

② Link alert - alert

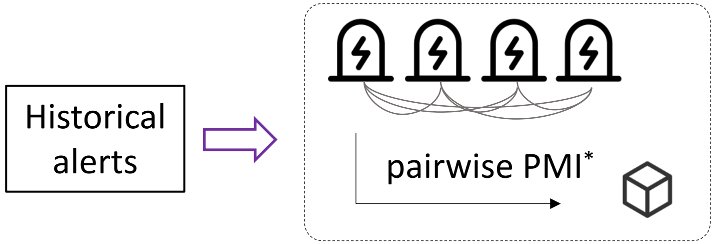


Overall Framework of iPACK

① Reduce redundant alerts

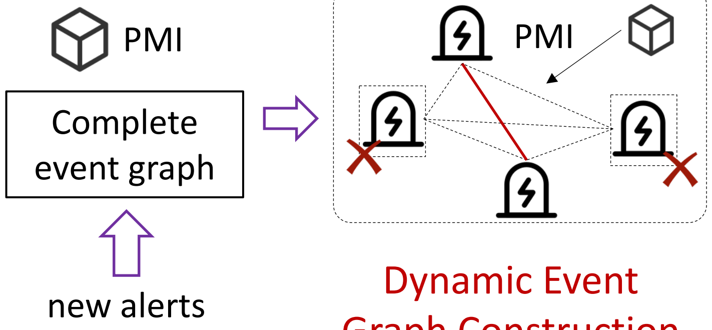
② Link alert - alert

How historical events correlate with each other?



Static Event Relation Learning (offline)

Prune events that correlate with most of other events.



Dynamic Event Graph Construction (online)

Graph-based Incident Profiling (GIP)

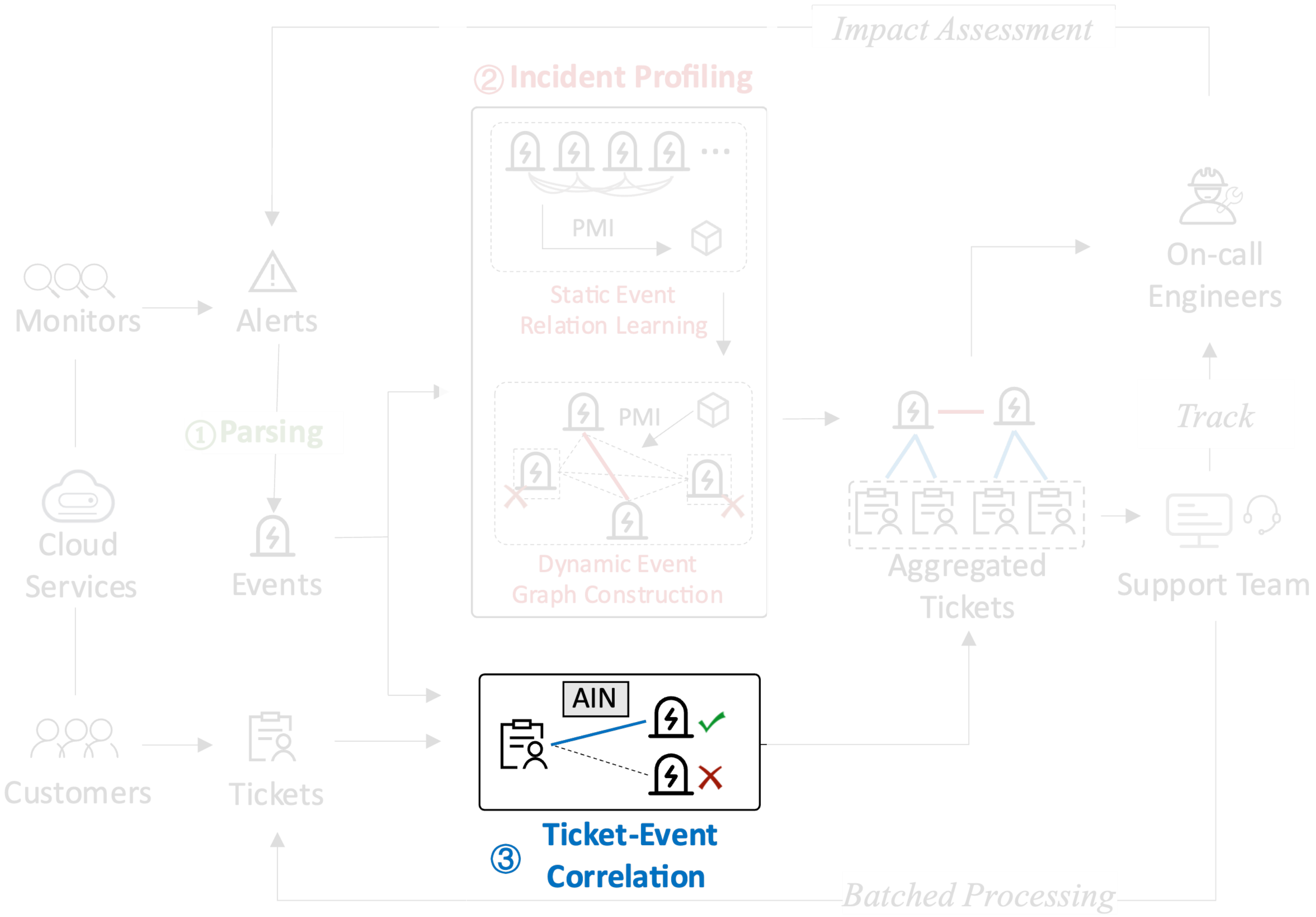
PMI: Pointwise mutual information

Overall Framework of iPACK

① Reduce redundant alerts

② Link alert - alert

③ Link ticket - alert

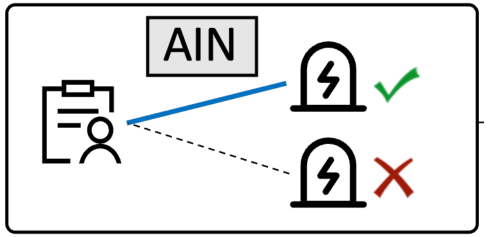


Overall Framework of iPACK

① Reduce redundant alerts

② Link alert - alert

③ Link ticket - alert



③ Ticket-Event Correlation

AIN: Attentive Interaction Network

⚠ Alert: 21456282 Status: Active

Title: Synthetics-API-Latency [PUT_WestUS] is degraded in last 20 mins.

Creation Time: 2022/7/25 12:14:26 Region: West US

Owning Service: Kubernetes Severity: Medium

Owning Component: Kubernetes\Scheduler Monitor ID:

Ticket: 2022072505 Status: Open

Summary: Error deploying the container. Region: West US

Creation Time: 2022/7/25 15:34:42 Product Name: Kubernetes

Category: Kubernetes\container creation\cannot create

feature combinations

feature1 (100) feature2 (1000) feature1+2 (100 * 1000)

OneHot: 1 0 0 0 0 ... 0 1 0 0 ... 0 0 0 0 0 1 ...

Decomposition: ⊙

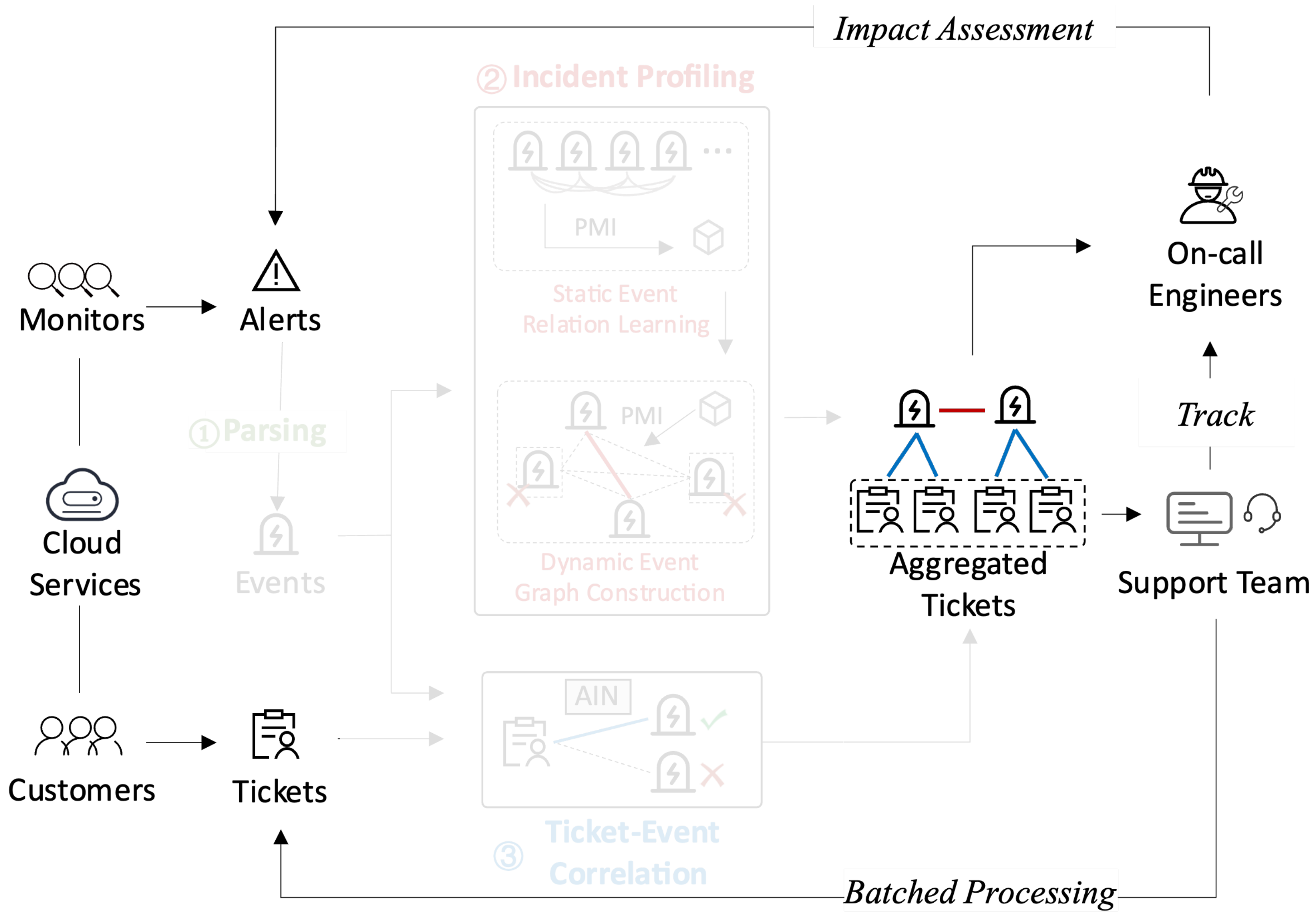
embedding1 *embedding2* *interaction*

Overall Framework of iPACK

① Reduce redundant alerts

② Link alert - alert

③ Link ticket - alert



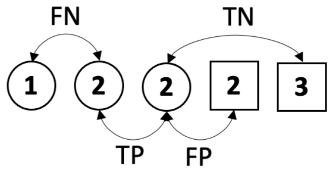
Evaluation

- Dataset

Dataset	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	Total
# Services	49	57	51	81
# Incidents	462	579	642	1,575
# Tickets/Incident	23∼275	36∼292	25∼95	23∼292
# Alerts	398,735	409,590	445,089	1,253,414

Three regions from Azure

- Metrics
  - How well a method can cluster duplicate tickets together?
    - Rand index-based precision, recall and F1



Shape: prediction
Number: ground truth

$$precision = \frac{TP}{TP+FP}, recall = \frac{TP}{TP+FN}, \text{and } F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Evaluation

- Overall effectiveness of iPACK

Methods	Dataset $\mathcal{A}$ Precision	Dataset $\mathcal{A}$ Recall	Dataset $\mathcal{A}$ **F1 score**	Dataset $\mathcal{B}$ Precision	Dataset $\mathcal{B}$ Recall	Dataset $\mathcal{B}$ **F1 score**	Dataset $\mathcal{C}$ Precision	Dataset $\mathcal{C}$ Recall	Dataset $\mathcal{C}$ **F1 score**
Categorization	0.930	0.205	0.336	0.943	0.373	0.535	0.925	0.207	0.338
iFeedback	0.901	0.590	<u>0.713</u>	0.876	0.473	0.614	0.886	0.626	0.733
LWE	0.862	0.453	0.594	0.824	0.515	0.634	0.861	0.672	<u>0.755</u>
BERT	0.884	0.587	0.705	0.854	0.710	<u>0.775</u>	0.843	0.629	0.720
LinkCM	0.931	0.507	0.657	0.892	0.538	0.671	0.901	0.628	0.740
LinkCM w/ GIP	0.900	0.685	0.778	0.886	0.756	0.816	0.899	0.809	0.852
iPACK	0.912	0.960	**0.935**	0.882	0.861	**0.871**	0.899	0.888	**0.894**

Observation 1: SOTA semantic-based baselines achieve high precision but low recall

Evaluation

- Overall effectiveness of iPACK

Methods	Dataset $\mathcal{A}$			Dataset $\mathcal{B}$			Dataset $\mathcal{C}$		
	Precision	Recall	**F1 score**	Precision	Recall	**F1 score**	Precision	Recall	**F1 score**
Categorization	0.930	0.205	0.336	0.943	0.373	0.535	0.925	0.207	0.338
iFeedback	0.901	0.590	0.713	0.876	0.473	0.614	0.886	0.626	0.733
LWE	0.862	0.453	0.594	0.824	0.515	0.634	0.861	0.672	0.755
BERT	0.884	0.587	0.705	0.854	0.710	0.775	0.843	0.629	0.720
LinkCM	0.931	0.507	0.657	0.892	0.538	0.671	0.901	0.628	0.740
LinkCM w/ GIP	0.900	0.685	0.778	0.886	0.756	0.816	0.899	0.809	0.852
iPACK	0.912	0.960	**0.935**	0.882	0.861	**0.871**	0.899	0.888	**0.894**

Observation 1: SOTA semantic-based baselines achieve high precision but low recall

Observation 2: iPACK slightly sacrifices precision and achieves best overall performance

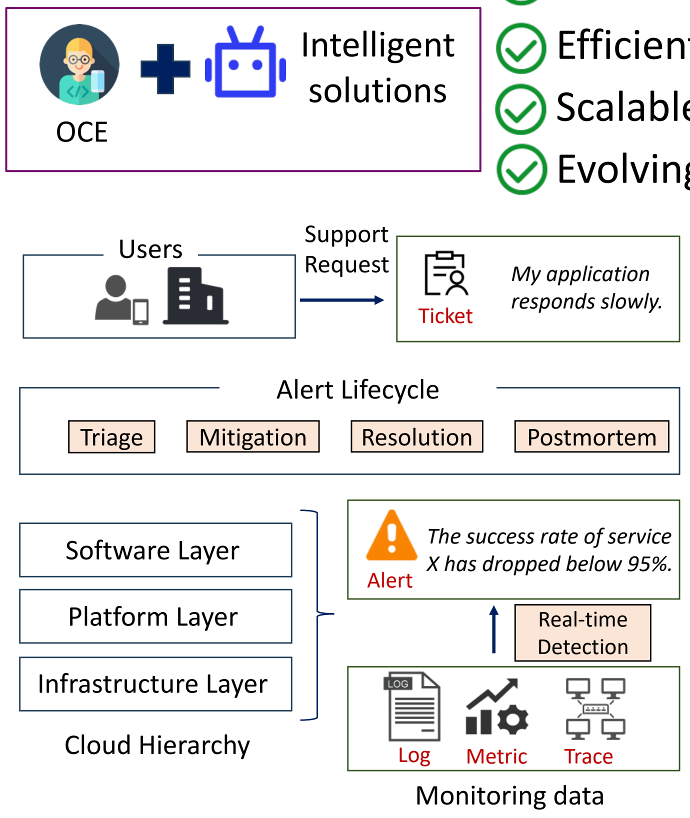
Summary of ③ iPACK

- Duplicate ticket in cloud systems can poccess **semantically different content** caused by inter-dependent services, making existing work ineffective.

- We propose iPACK to **introduce alerts** to facilitate grouping duplicate tickets.
  - Incident Profiling: alert – alert linking
  - Attentive Intraction Network: alert – ticket linking

- iPACK outperforms existing state-of-the-art solutions by **12.4% ~ 31.2%** across three industial datasets collected from **Azure.**

Thesis Conclusion



Reliability Management

- Automated
- Efficient
- Scalable
- Evolving

① Large Scale

② Complicated Dependencies

③ Fast Evolving

④ Limited Observations

Thesis Contribution

① Sealog

Scalable and adaptive log-based anomaly detection

② Prism

Improving observability across different layers of cloud hierarchy

③ iPACK

Correlating tickets and alerts for more comprehensive ticket deduplication

Future work: Towards Autonomous Cloud Systems

High-quality data — Unified abstraction → **Intelligent solutions**

Goal: comprehensively and precisely capture system runtime information		Goal: Unify and leverage cloud knowledge to automate various Ops tasks.	
Where/What to Monitor	How to Monitor	How to Analyse	How to Decide
Postmortem Analysis	Monitor Rules Generation	Interactive ChatOps	Automatic Mitigation
KPI Design (aggregation)	Monitor Recommendation	Cloud Q&A (internal & external)	Root Cause Analysis
Full Link Topology	Monitor Optimization	Interpretable Multi-modal Analysis	Impact Scoping

LLM-powered Operation Agent

- Domain knowledge adaption
- Tool Learning
- Prompt Optimization
- Self-debugging

Human Feedback

Publication list

- Zhihan Jiang, **Jinyang Liu**, Junjie Huang, Yichen Li, Yintong Huo, Jiazhen Gu, Zhuangbin Chen, Jieming Zhu and Michael R. Lyu. “LILAC: Log Parsing using LLMs with Adaptive Parsing Cache.” In Proceedings of the 32rd International Conference on Software Engineering (**FSE 2024**).
- Zhihan Jiang, **Jinyang Liu**, Junjie Huang, Yichen Li, Yintong Huo, Jiazhen Gu, Zhuangbin Chen, Jieming Zhu and Michael R. Lyu. “A Large-scale Evaluation for Log Parsing Techniques: How Far are We?.” In Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis (**ISSTA 2024**).
- Junjie Huang, **Jinyang Liu**, Zhuangbin Chen, Zhihan Jiang, Yichen Li, Jiazhen Gu, Cong Feng, Zengyin Yang, Yongqiang Yang, and Michael R. Lyu. “FaultProfIT: Hierarchical Fault Profiling of Incident Tickets in Large-scale Cloud Systems.” In Proceedings of the 46th International Conference on Software Engineering, Software Engineering in Practice track (**ICSE-SEIP 2024**).
- Jinxi Kuang, **Jinyang Liu**, Junjie Huang, Renyi Zhong, Jiazhen Gu, Lan Yu, Rui Tan, Zengyin Yang, Michael R. Lyu. “Knowledge-aware Alert Aggregation in Large-scale Cloud Systems: a Hybrid Approach.” In Proceedings of the 46th International Conference on Software Engineering, Software Engineering in Practice track (**ICSE-SEIP 2024**).
- **Jinyang Liu**, Junjie Huang, Yintong Huo, Zhihan Jiang, Jiazhen Gu, Zhuangbin Chen, Cong Feng, Minzhi Yan, Michael R Lyu. “Scalable and Adaptive Log-based Anomaly Detection with Expert in the Loop.” (**under review, ICSE 2025**).
- **Jinyang Liu**, Shilin He, Zhuangbin Chen, Liqun Li, Yu Kang, Xu Zhang, Pinjia He, Hongyu Zhang, Qingwei Lin, Zhangwei Xu, Saravan Rajmohan, Dongmei Zhang, Michael Lyu. “Incident-aware Duplicate Ticket Aggregation for Cloud Systems.” In Proceedings of the 45h International Conference on Software and Engineering (**ICSE 2023**).
- **Jinyang Liu**, Tianyi Yang, Zhuangbin Chen, Yuxin Su, Cong Feng, Zengyin Yang, Michael R. Lyu. “Practical Anomaly Detection over Multivariate Monitoring Metrics for Online Services.” In Proceedings of the 34th IEEE International Symposium on Software Reliability Engineering (**ISSRE 2023**).
- **Jinyang Liu***, Zhihan Jiang*, Jiazhen Gu, Junjie Huang, Zhuangbin Chen, Cong Feng, Zengyin Yang, Yongqiang Yang, Michael R. Lyu. “Prism: Revealing Hidden Functional Clusters from Massive Instances in Cloud Systems.” In Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering (**ASE 2023**).

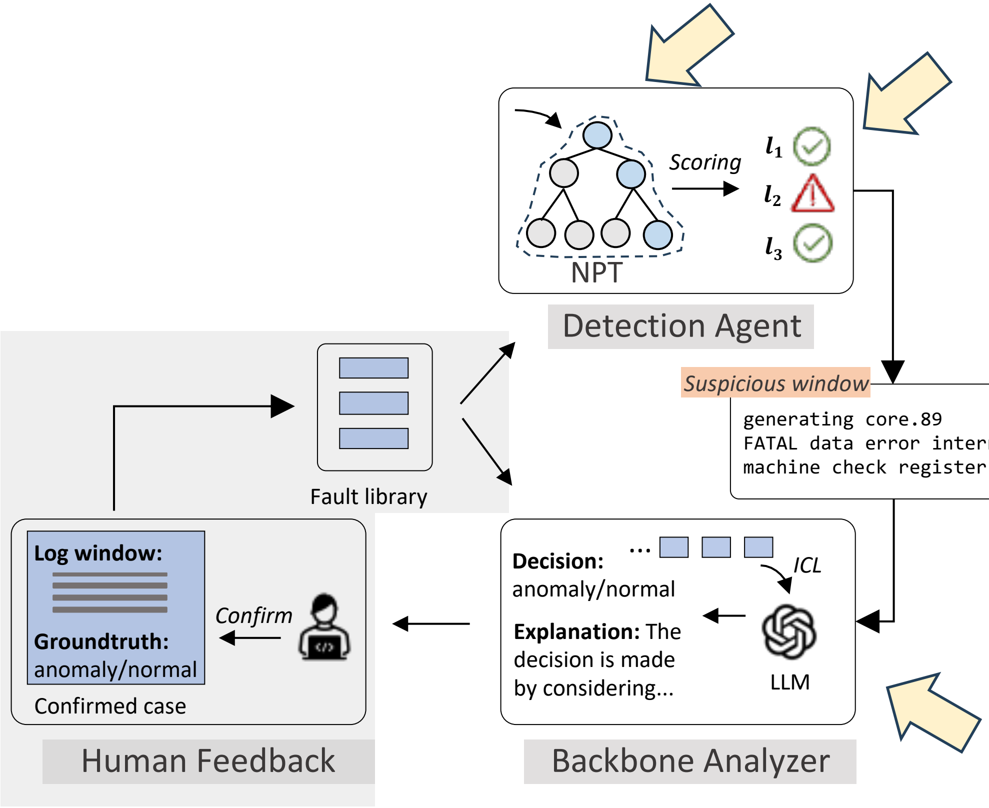
Publication list

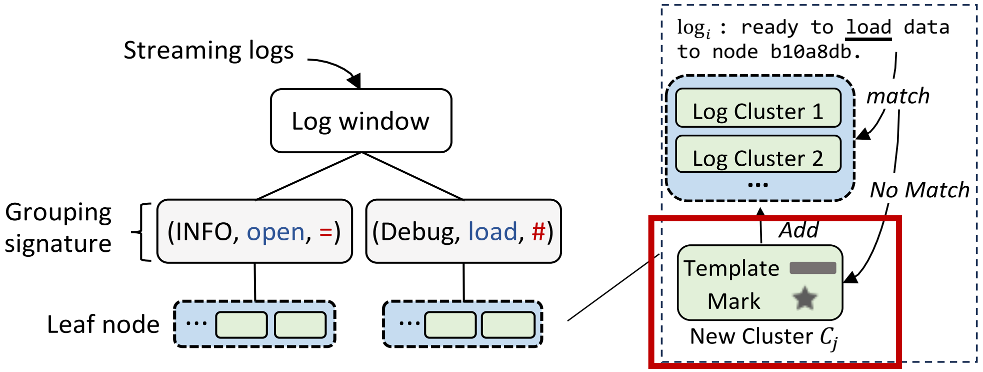
- Jieming Zhu, Shilin He, Pinjia He, **Jinyang Liu**, Michael R. Lyu. “Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics.” In Proceedings of the 34th IEEE International Symposium on Software Reliability Engineering (**ISSRE 2023**).
- Zhuangbin Chen, **Jinyang Liu**, Yuxin Su, Hongyu Zhang, Xiao Ling, Michael R. Lyu. “Adaptive Performance Anomaly Detection for Online Service Systems via Pattern Sketching.” In Proceedings of the 44th International Conference on Software and Engineering (**ICSE 2022**).
- Yichen Li, Xu Zhang, Shilin He, Zhuangbin Chen, Yu Kang, **Jinyang Liu**, Liqun Li, Yingnong Dang, Feng Gao, Zhangwei Xu, Saravan Rajmohan, Qingwei Lin, Dongmei Zhang, Michael R. Lyu. “An Intelligent Framework for Timely, Accurate, and Comprehensive Cloud Incident Detection.” ACM SIGOPS Operating Systems Review, 2022.
- Zhuangbin Chen, **Jinyang Liu**, Yuxin Su, Hongyu Zhang, Xuemin Wen, et al. Graph-based Incident Aggregation for Large-Scale Online Service Systems.” In Proceedings of The 36th IEEE/ACM International Conference on Automated Software Engineering (**ASE 2021**).
- **Jinyang Liu**, Jieming Zhu, Shilin He, Pinjia He, Zibin Zheng, and Michael R. Lyu. “Logzip: Extracting Hidden Structures via Iterative Clustering for Log Compression.” In Proceedings of the 34th IEEE/ACM International Conference on Au- tomated Software Engineering (**ASE 2019**).
- Jieming Zhu, Shilin He, **Jinyang Liu**, Pinjia He, Qi Xie, Zibin Zheng, Michael R. Lyu. Tools and Benchmarks for Auto- mated Log Parsing.” In Proceedings of the 41st International Conference on Software Engineering, Software Engineering in Practice track (**ICSE-SEIP 2019**).

Thank you!
Q & A

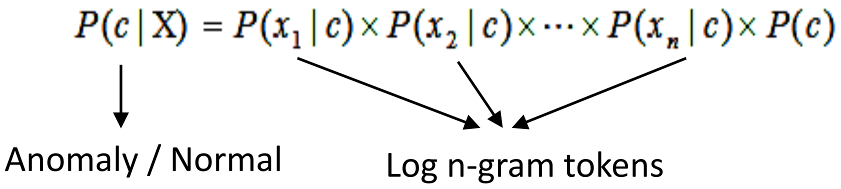
Backbone analyzer of Sealog

- Utilizing confirmed cases as feedback

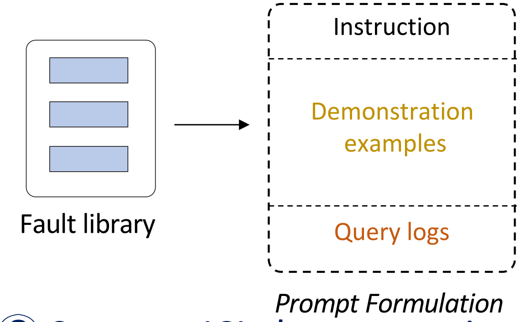




① Mark templates with labels



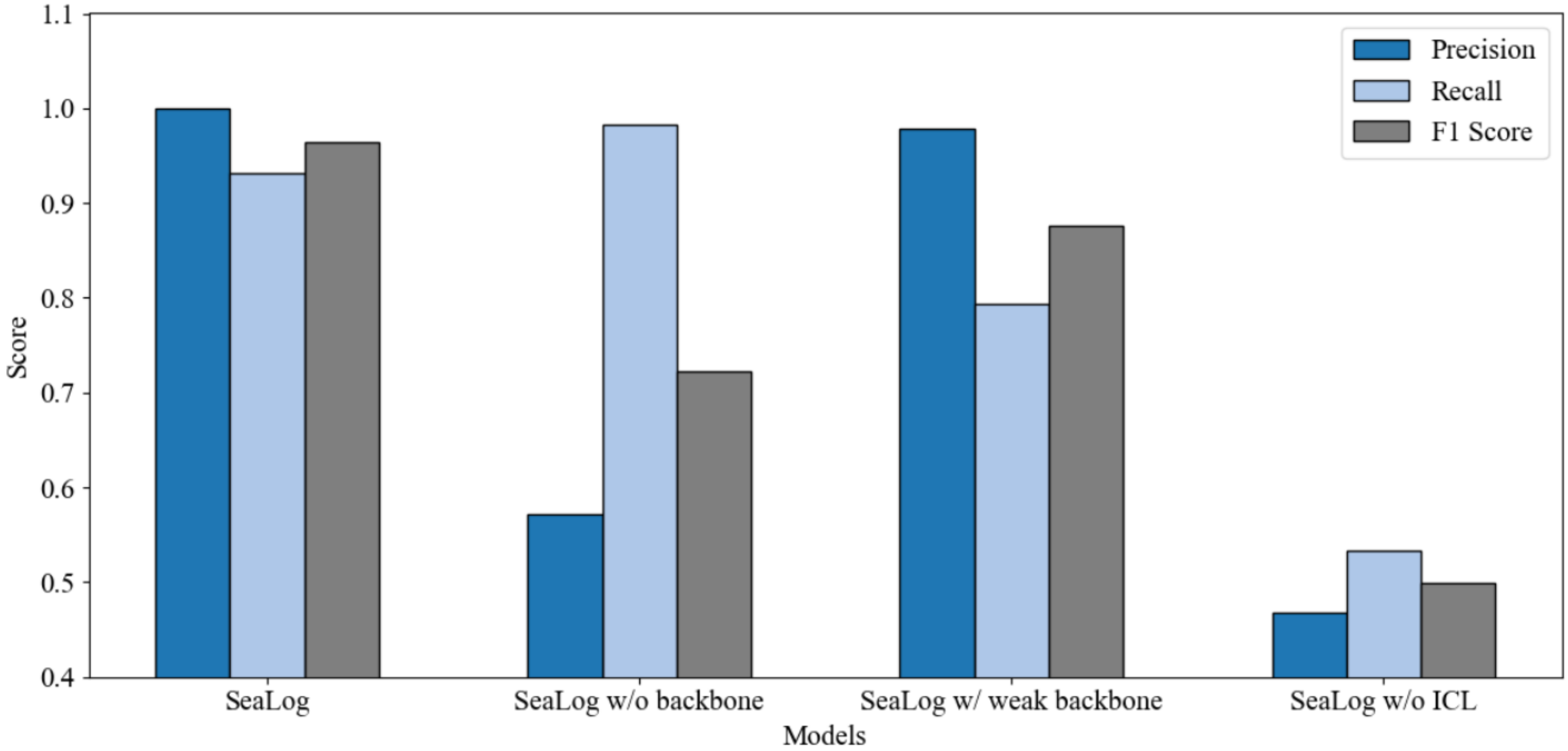
② Continuously update statistics



③ Serve as ICL demonstrations

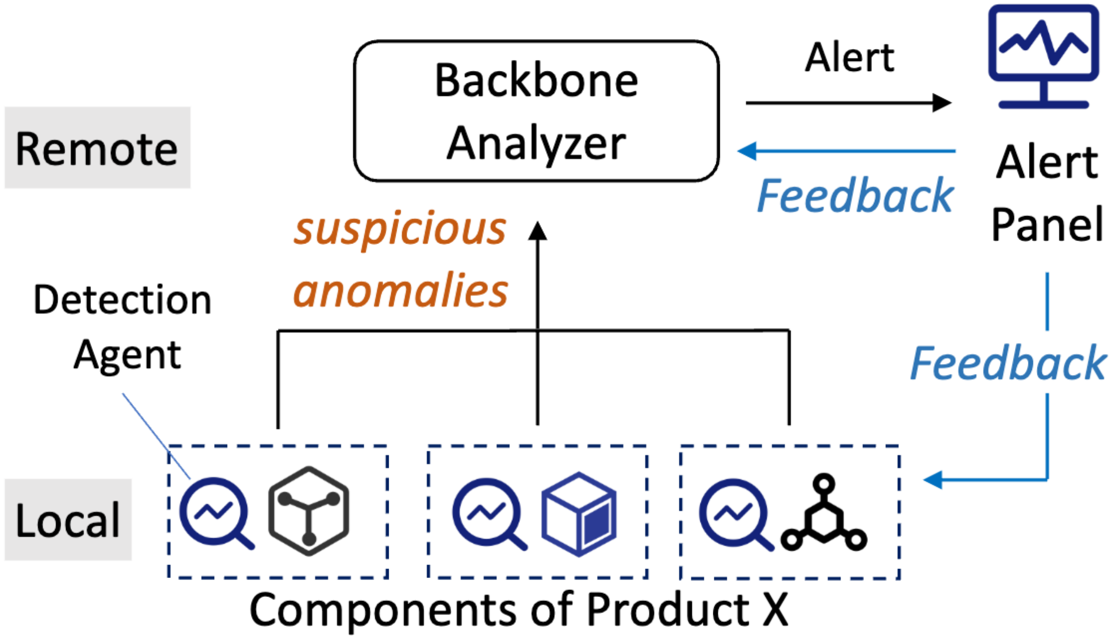
Evaluation

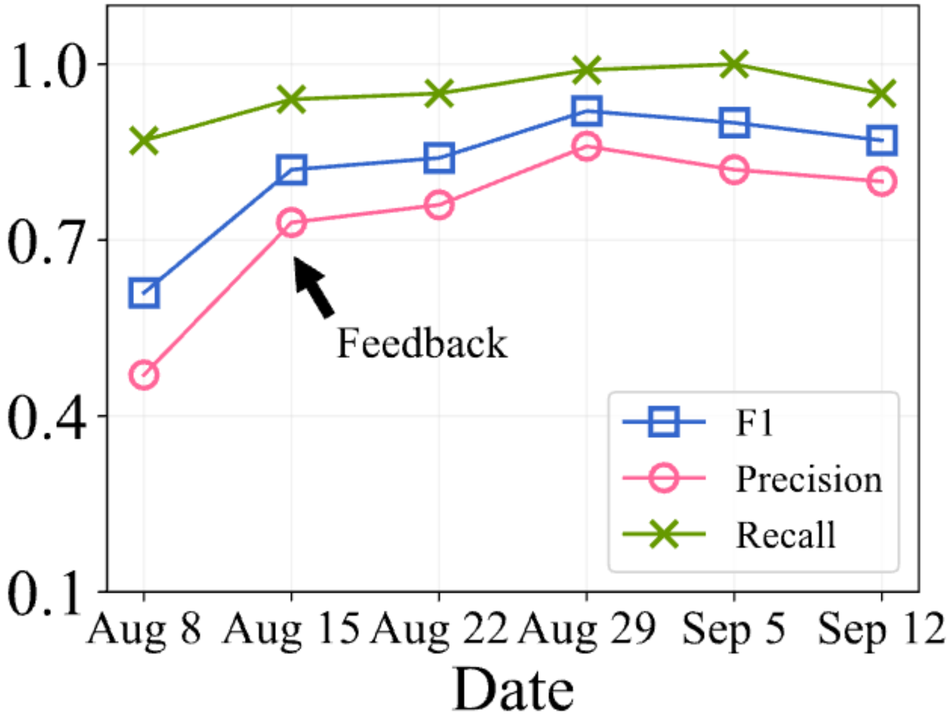
- Ablation study of Sealog



Industry deployment

- Deployment in Huawei Cloud





Analysis case of Sealog

Please determine if the given log messages indicate a system run-time anomaly or not. In the following, some similar examples are provided for reference, you should compare the given log messages with them and make your own decision.

Query:
"User login attempt took longer than expected, <*> seconds."

Examples:
Input: "Unable to reach the authentication server. Timeout occurred after <*> seconds." Label: Anomaly

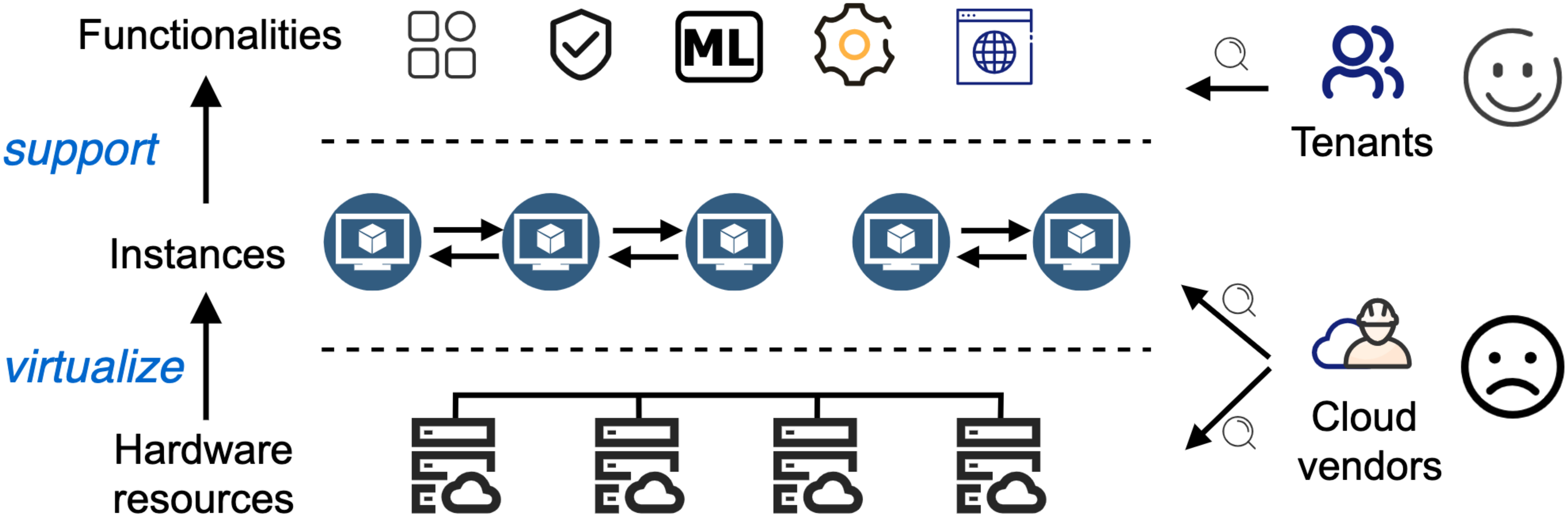
The output MUST be in standard JSON format and MUST consist of TWO keys: 'prediction' and 'analysis' .
"Prediction": <you should choose one of from 'normal' or 'anomaly' according to your analysis, do not use other words.>
"Analysis": <your analysis for the given log messages based on its semantics>
```

```

Output of backbone analyzer:

```
{
  "Prediction": "anomaly",
  "Analysis": "The log message 'User login attempt took longer than expected, <*> seconds.' indicates that the login process is experiencing delays beyond the expected duration. This is similar to the example where a timeout occurs after a certain number of seconds, which is labeled as an anomaly. Therefore, the prolonged login attempt time suggests a potential issue in the system's performance."
}
```

Cloud Infrastructure



Virtualization reduces cloud observability for cloud vendors during maintenance tasks.

Our solution: two-stage linking

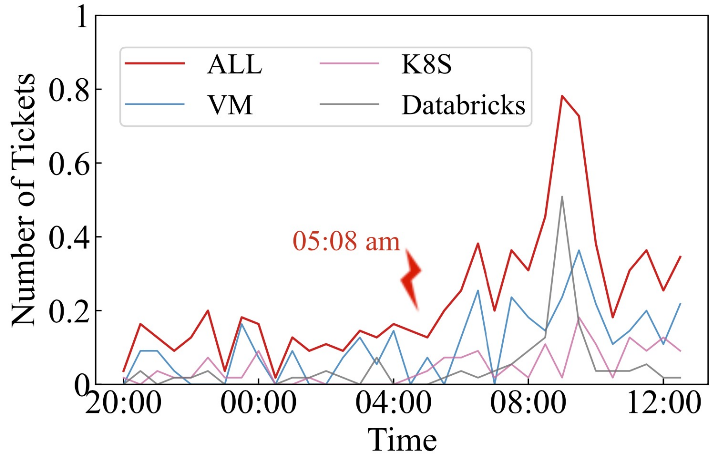
- Intuition: first identify impacted services, then associated tickets

Depend on

incident

Disk Resource Provider

Virtual Machine
Kubernetes
Databricks



① service | service

Service	Tickets	
	Category	Summary
VM	VM/Scale Update	t_1: Virtual machine scale sets resize issue.
	VM/VM Start	t_2: Server did not start on time.
Databricks	Databricks/Job Issue	t_3: Unable to open cluster of Databricks.
	Databricks/Cluster Launch	t_4: Unable to provision clusters.
K8S	K8S/Cluster Update	t_5: Unable to autoscale.
	K8S/Cluster Update	t_6: Cannot upgrade node pool, stuck.

② service | tickets

Our solution: two-stage linking

- How to know what services are affected by an incident?

Monitoring rules

If [latency] of [API] > [threshold]

⚠ **Alert**

⚠ Alert: 21456282 Status: Active

Title: Synthetics-API-Latency [PUT_WestUS] is degraded in last 20 mins.

Creation Time: 2022/7/25 12:14:26 Region: West US

Owning Service: Kubernetes Severity: Medium

Owning Component: Kubernetes\Scheduler Monitor ID: 68ba52c9f

time locations issue Severity

incident ① service | service ② service | tickets

incident ① alert | alert ② alert | tickets

Methodology

• Alert Parsing (reduce redundancy)

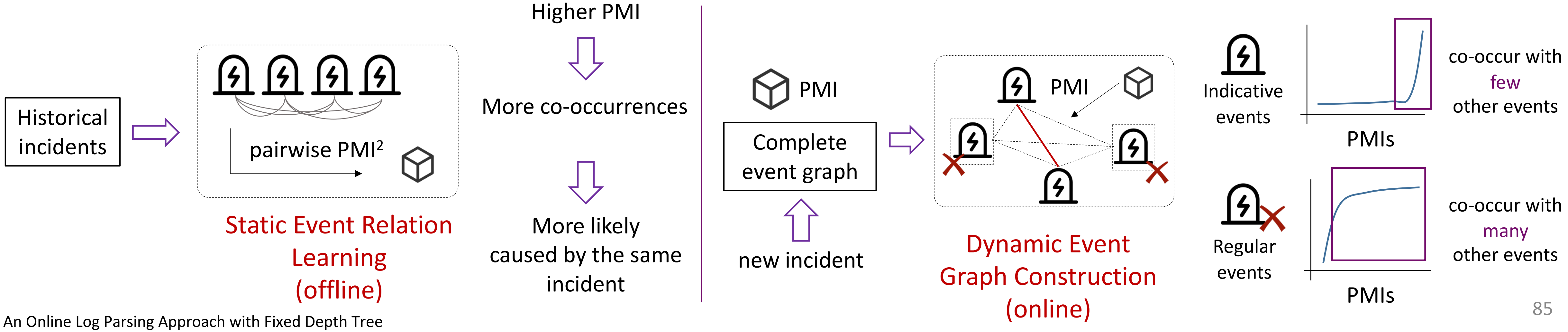
Redundant Alert

⚠ VMStart Failures exceed 100 times
⚠ VMStart Failures exceed 150 times
⚠ VMStart Failures exceed 200 times
⚠ VMStart Failures exceed 250 times

Drain[1] (log parser) ⇨

Events: VMStart Failures exceed <*> times

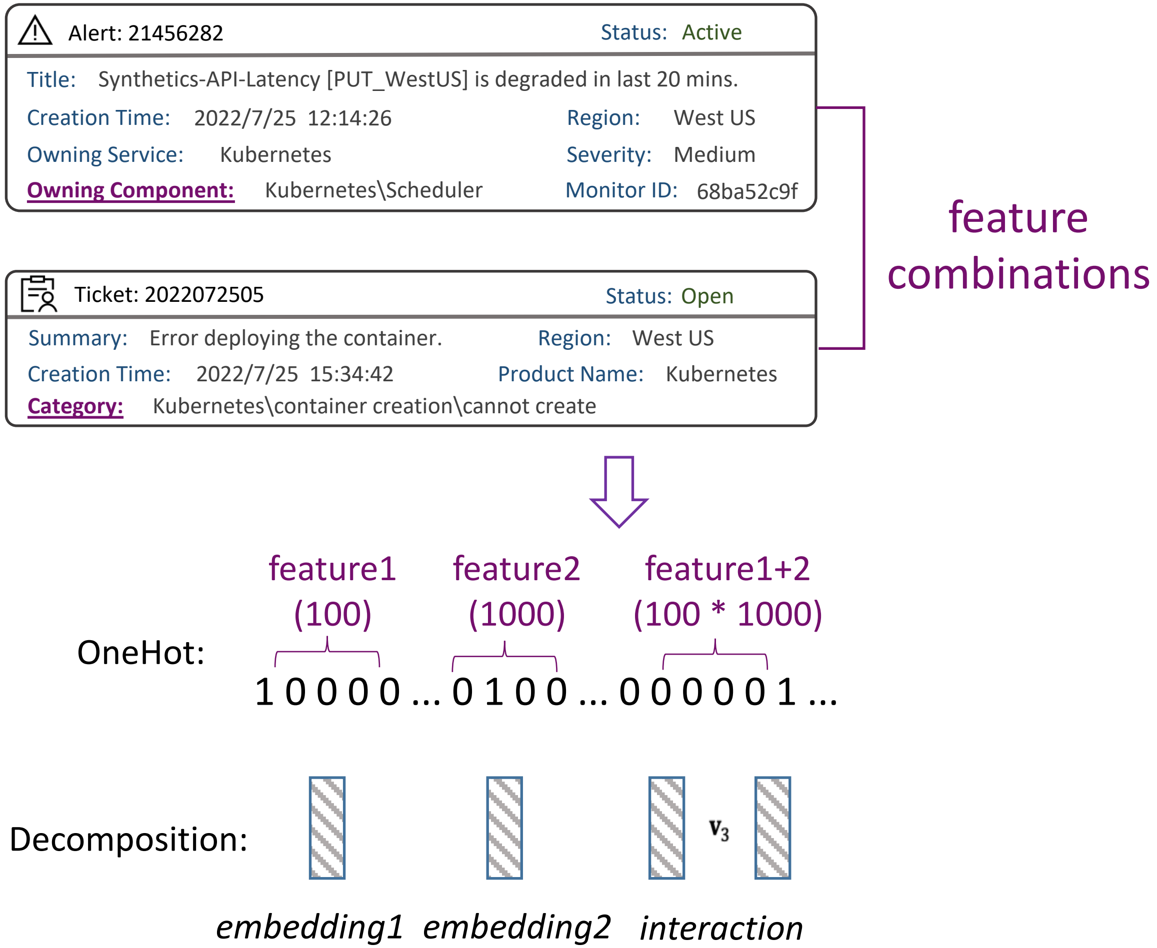
• Incident Profiling (reduce regular events & link indicative events)

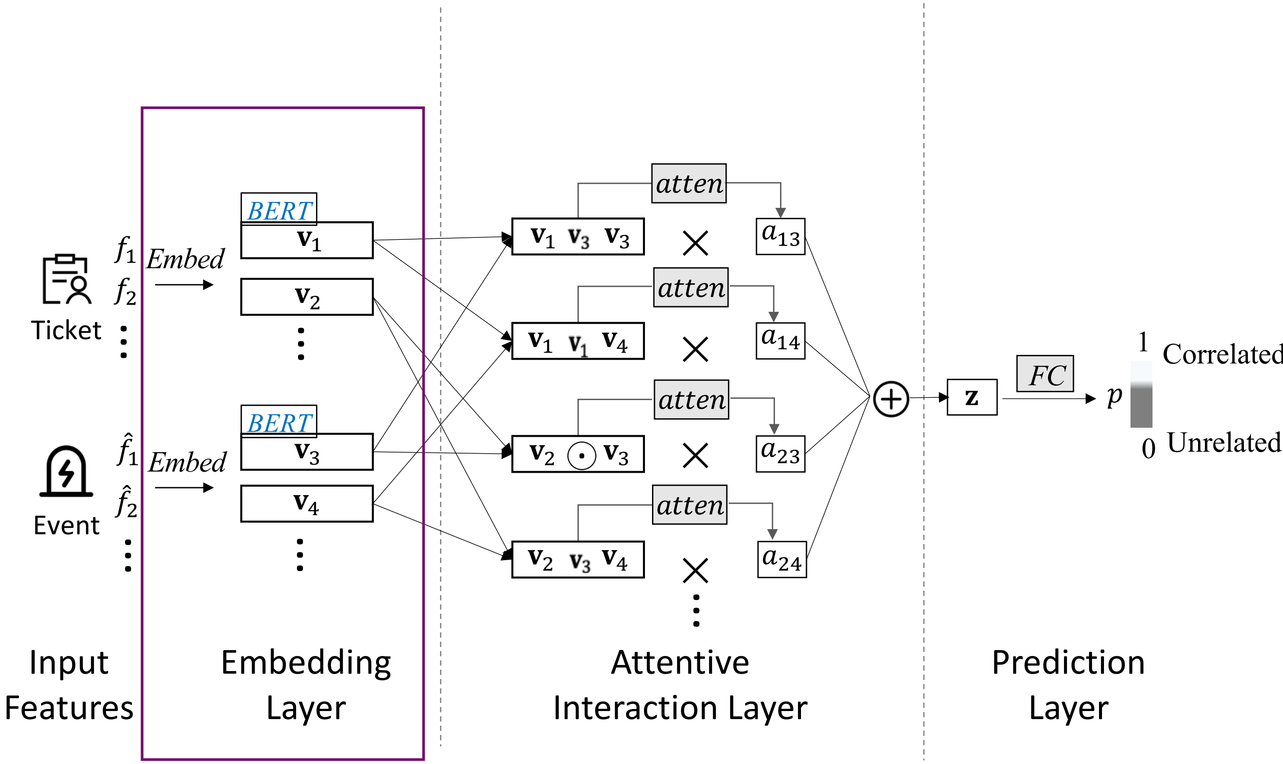


[1] Drain: An Online Log Parsing Approach with Fixed Depth Tree
[2] PMI: Pointwise mutual information

Methodology

- Ticket-Event Correlation (link tickets to events)





Attentive Interaction Network (AIN)

Evaluation

- Research Questions

RQ1: How effective is iPACK in aggregating duplicate tickets?

RQ2: How effective is AIN in correlating tickets and events?

RQ3: How does incident profiling impact the effectiveness of iPACK?

Evaluation

- RQ2: The Effectiveness of ticket-event correlation

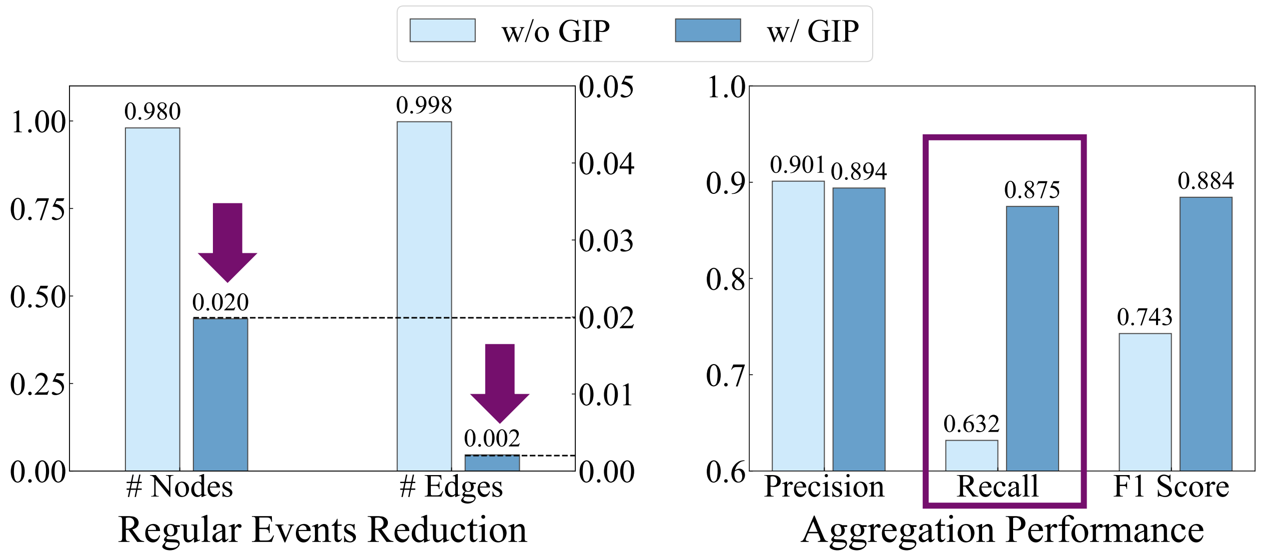
Models	Acc@1	Acc@2	Acc@3	Average
LR	0.519	0.657	0.733	0.636
SVM	0.332	0.409	0.493	0.411
RF	0.563	0.684	0.761	0.669
LightGBM	0.658	0.723	0.832	0.712
LinkCM	0.743	0.769	0.882	0.798
AIN w/o atten.	0.673	0.762	0.824	0.753
AIN	**0.817**	**0.907**	**0.936**	**0.887**
Δ(%)	+21.4%	+19.0%	+13.6%	+17.8%

Observation 1: AIN outperforms existing SOTA solutions

Observation 2: The attention module improves AIN by 13.6% ~ 21.4%

Evaluation

- RQ3: The Effectiveness of incident profiling



Observation 1: Only 2% of events are reserved after applying pruning

Observation 2: Incident Profiling mainly contributes to the Recall with an improvement from 0.632 to 0.743