Practical Root Cause Localization for Microservice Systems via Trace Analysis

Zeyan Li[1], Junjie Chen[2], Rui Jiao[1], Nengwen Zhao[1], Zhijun Wang[3], Shuwei Zhang[3], Yanjun Wu[3], Long Jiang[3], Leiqin Yan[3], Zikai Wang[4], Zhekang Chen[4], Wenchi Zhang[4], Xiaohui Nie[1], Kaixin Sui[4], Dan Pei[1]

[1] 清華大學 Tsinghua University

[2] 天津大學 Tianjin University

[3] 中国民生银行 CHINA MINSHENG BANK

[4] 必示 BIZSEER

June 26 2021. IWQoS 2021

Outline

Background

Approach

Evaluation

Summary

Background

Microservice Systems

Background

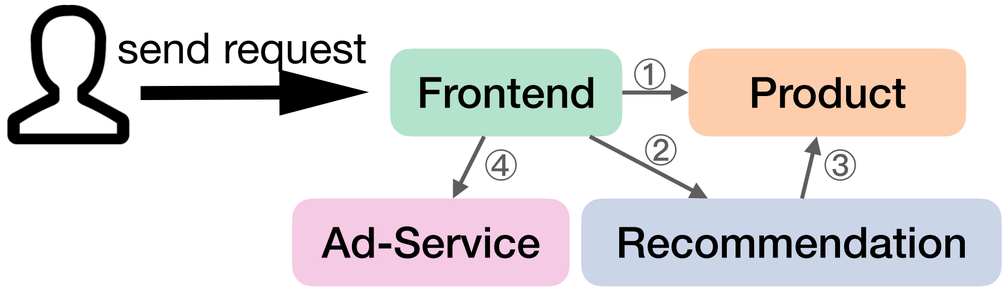
Microservice Systems

- Microservice architecture: the lasted trend to build systems

Background

Microservice Systems

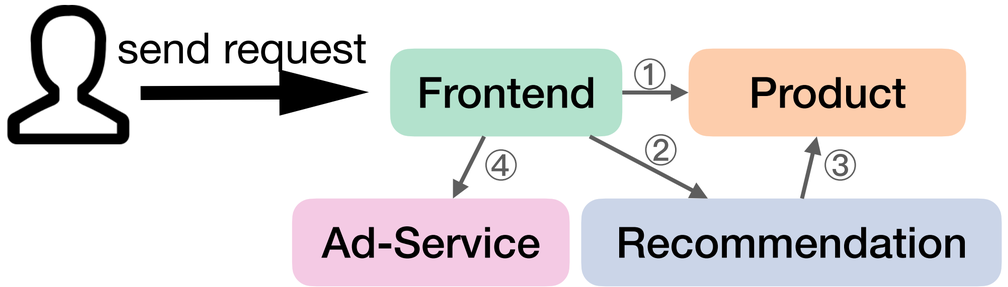
- Microservice architecture: the lasted trend to build systems
- What is microservice architecture?
  - Loosely coupled and independently deployed ‘micro’ services
  - Communicating with other services to realize a user request



Background

Microservice Systems

- Microservice architecture: the lasted trend to build systems
- What is microservice architecture?
  - Loosely coupled and independently deployed ‘micro’ services
  - Communicating with other services to realize a user request
- Benefits:
  - Fast delivery
  - Better scalability
  - Greater autonomy

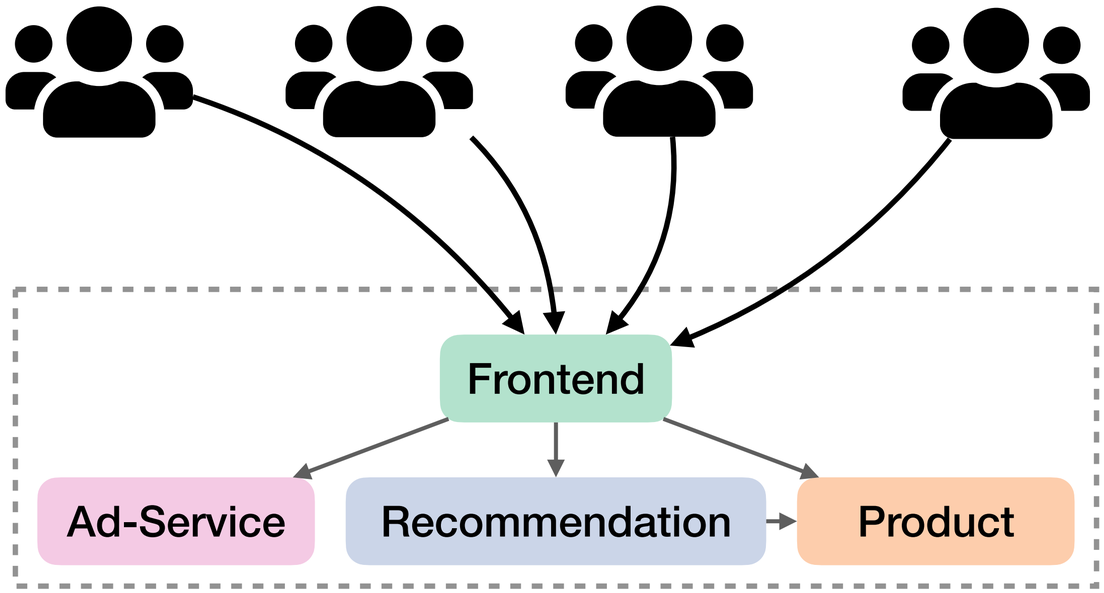


Background

Root-Cause Service Localization

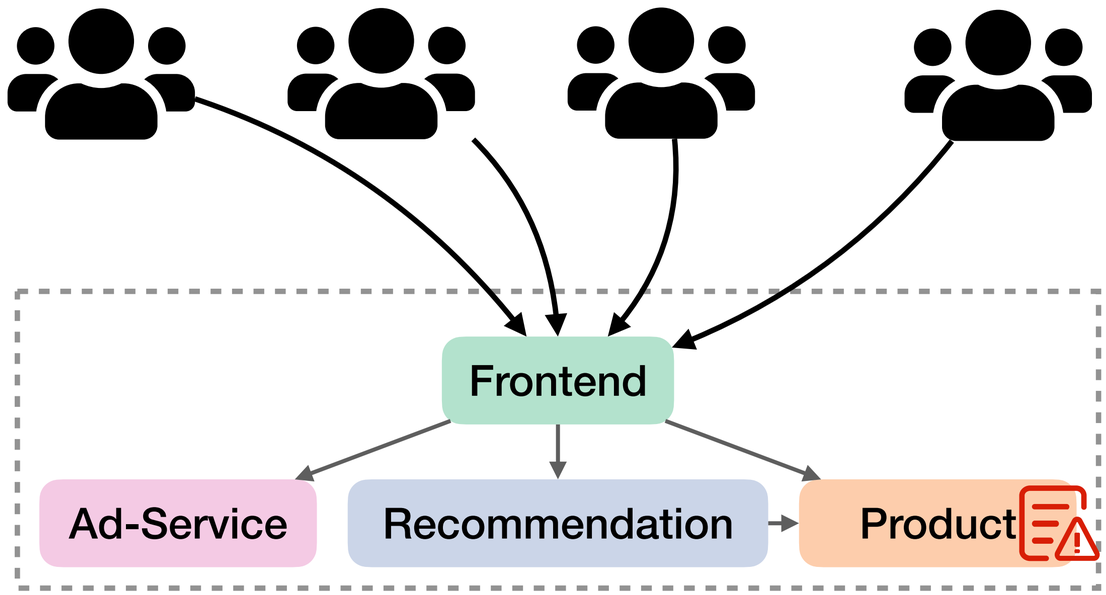
Background

Root-Cause Service Localization



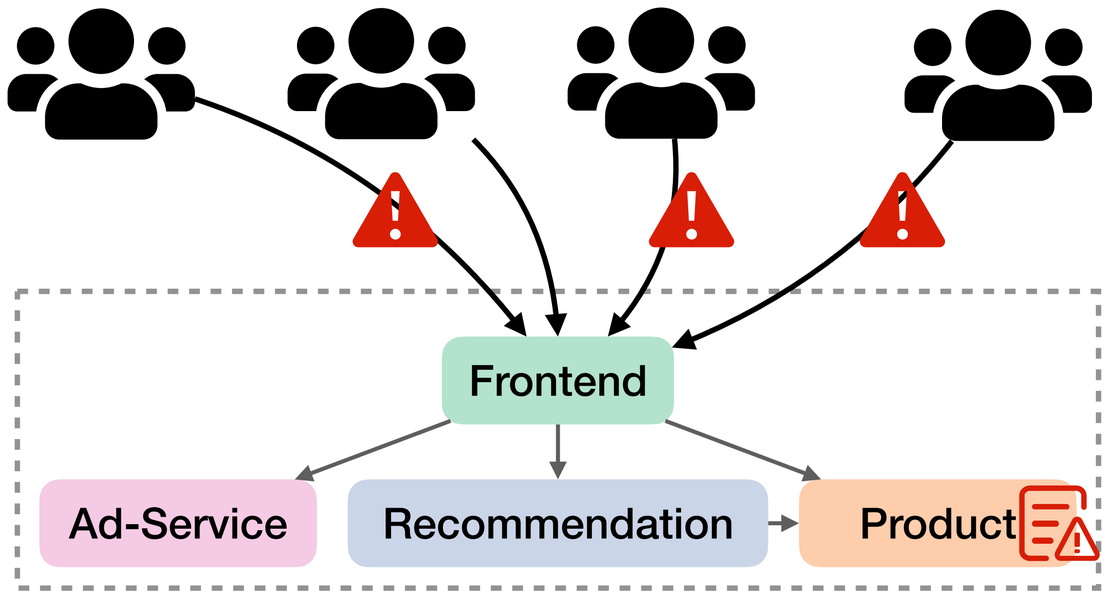
Background

Root-Cause Service Localization



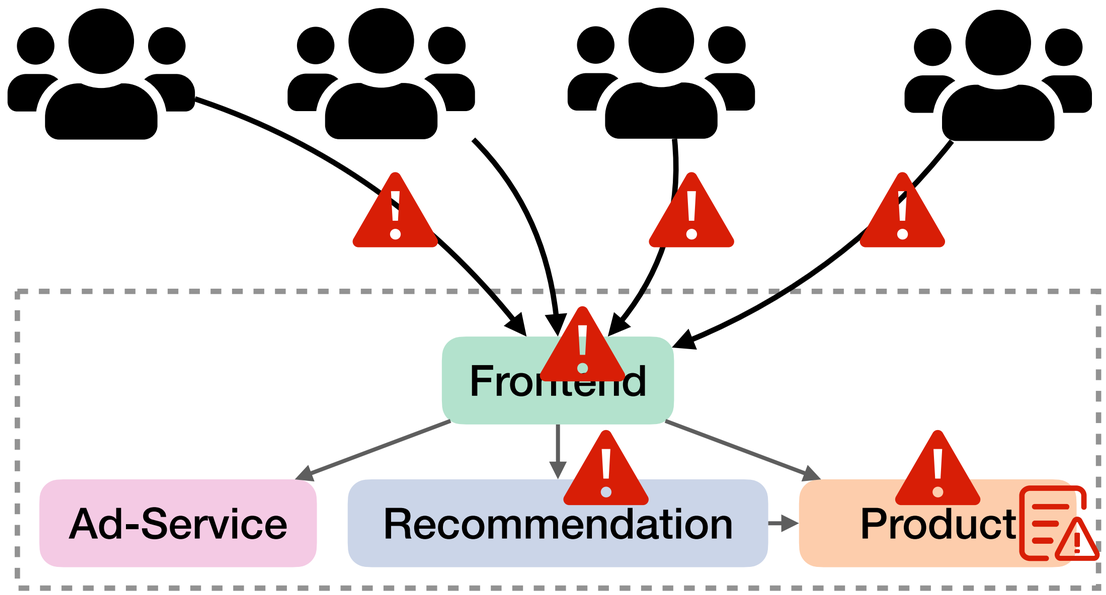
Background

Root-Cause Service Localization



Background

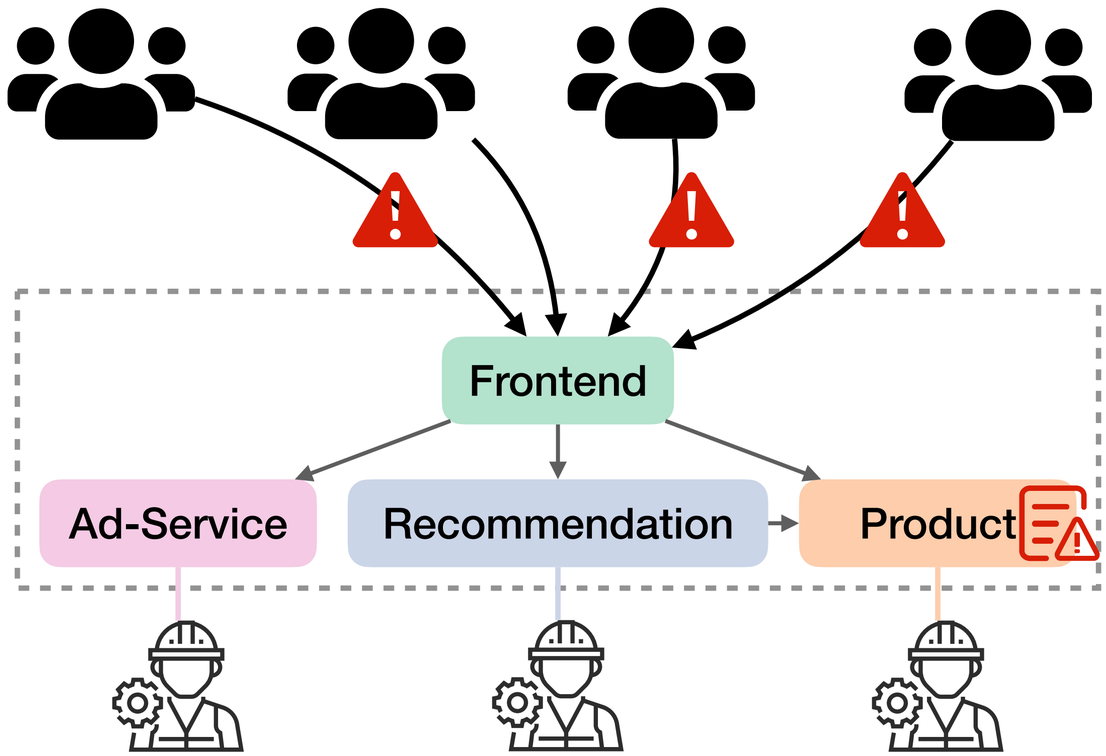
Root-Cause Service Localization



Many related services also behave abnormally due to the dependency among services

Background

Root-Cause Service Localization

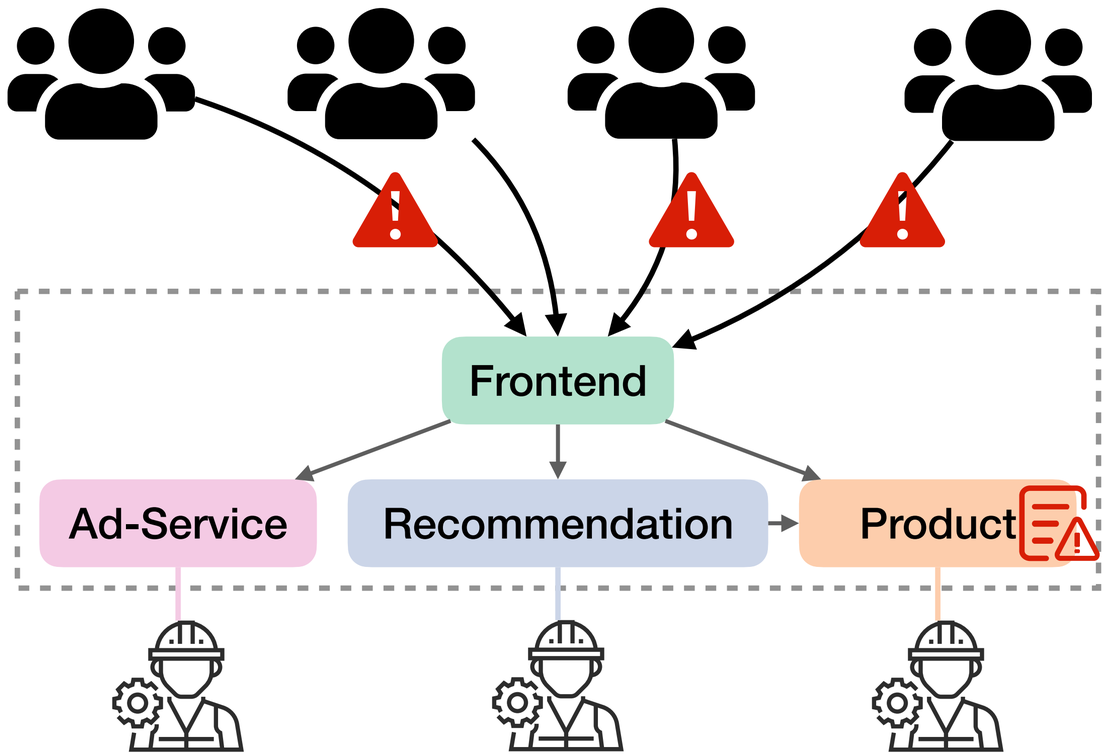


Many related services also behave abnormally due to the dependency among services

Different services are developed and maintained by different teams

Background

Root-Cause Service Localization



Many related services also behave abnormally due to the dependency among services

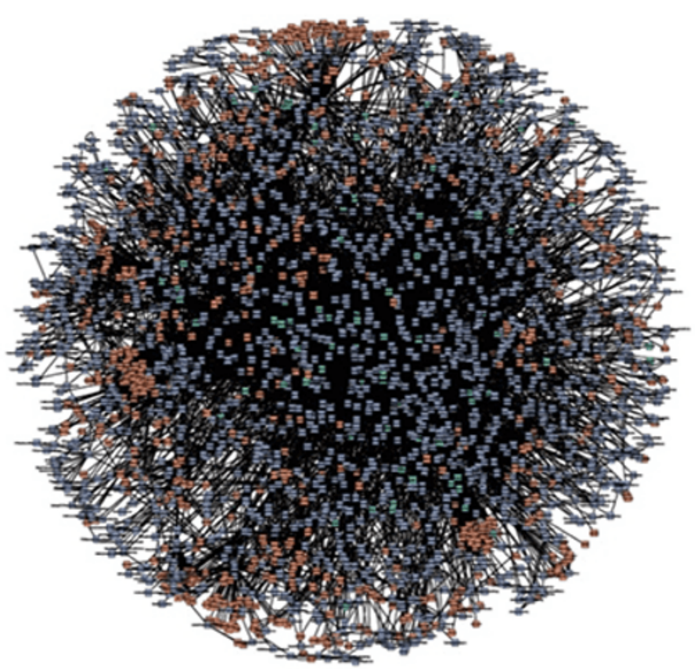
Different services are developed and maintained by different teams

When a fault happens, we need to localize the root-cause service at first before the corresponding team's further investigation

Background

Challenges in Microservice Systems

1. Complex dependencies among services



amazon.com

NETFLIX

Background

Challenges of Root Cause Localization in Microservice Systems

2. Dynamic runtime environments

Background

Challenges of Root Cause Localization in Microservice Systems

2. Dynamic runtime environments

COMPANY	DEPLOY FREQUENCY	DEPLOY LEAD TIME	RELIABILITY	CUSTOMER FEEDBACK
AMAZON	23.000/day	minutes	high	high
GOOGLE	5.500/day	minutes	high	high
NETFLIX	500/day	minutes	high	high
FACEBOOK	1/day	hours	high	high
TWITTER	3/week	hours	high	high
TYPICAL ENTERPRISE	once every 9 months	months or more	low/medium	low/medium

From "The Phoenix Project"

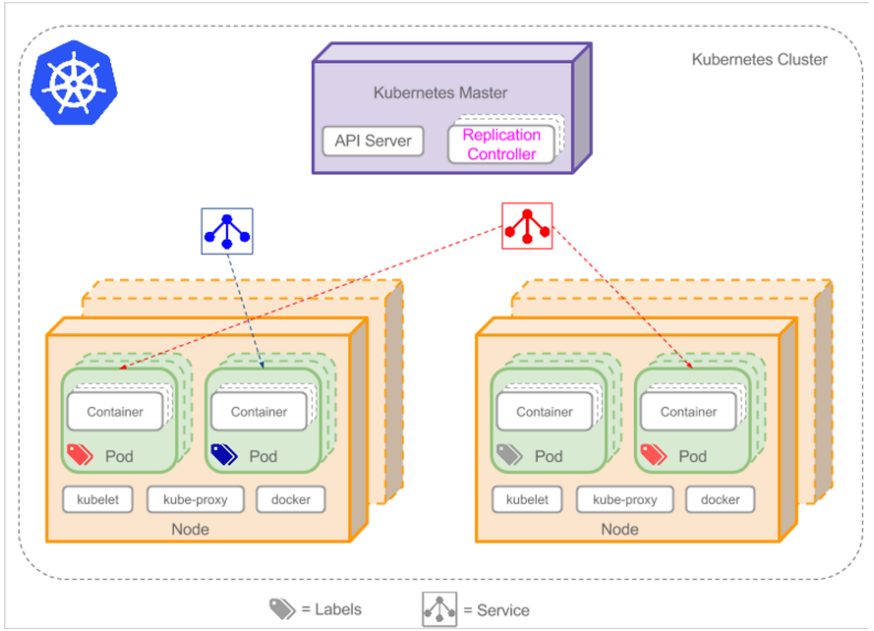
Background

Challenges of Root Cause Localization in Microservice Systems

2. Dynamic runtime environments

COMPANY	DEPLOY FREQUENCY	DEPLOY LEAD TIME	RELIABILITY	CUSTOMER FEEDBACK
AMAZON	23.000/day	minutes	high	high
GOOGLE	5.500/day	minutes	high	high
NETFLIX	500/day	minutes	high	high
FACEBOOK	1/day	hours	high	high
TWITTER	3/week	hours	high	high
TYPICAL ENTERPRISE	once every 9 months	months or more	low/medium	low/medium

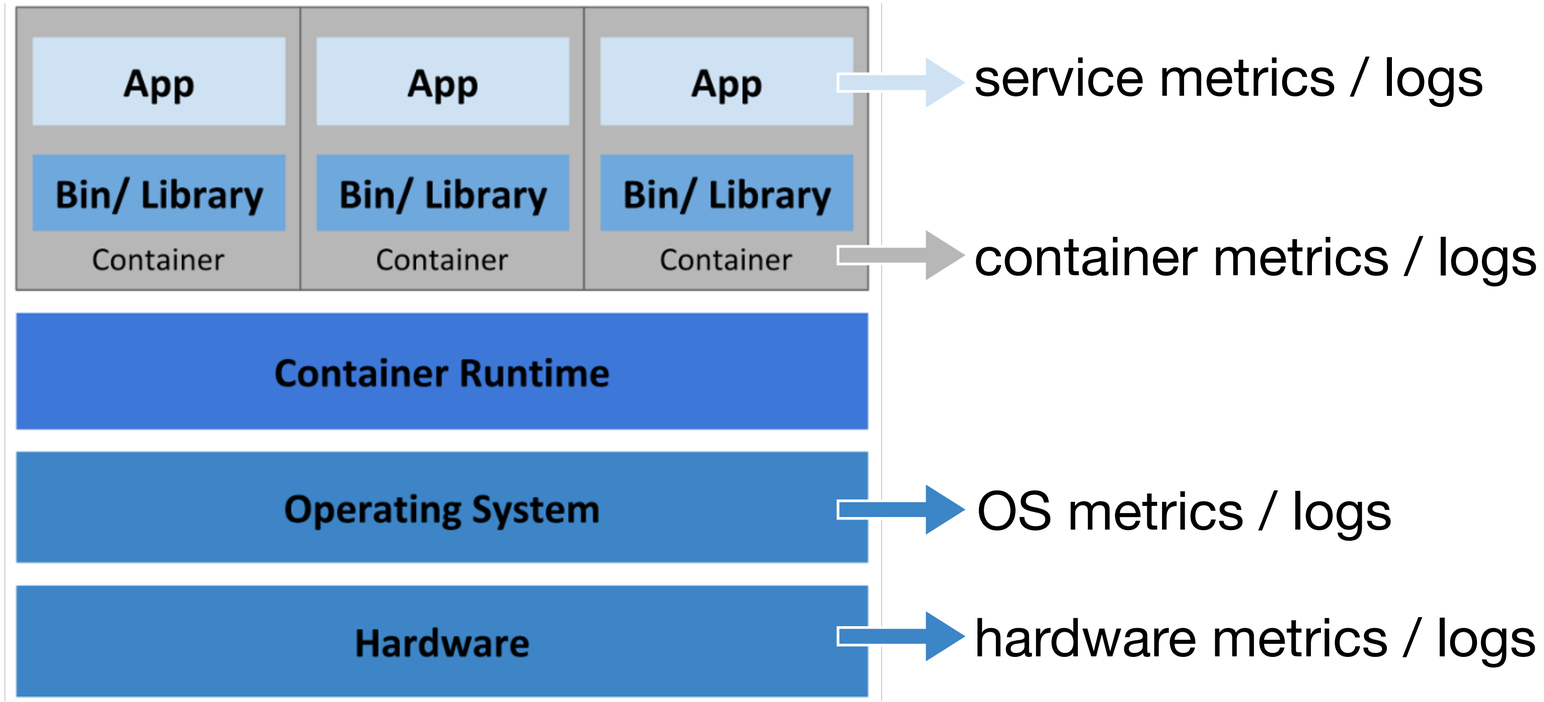
From "The Phoenix Project"



Background

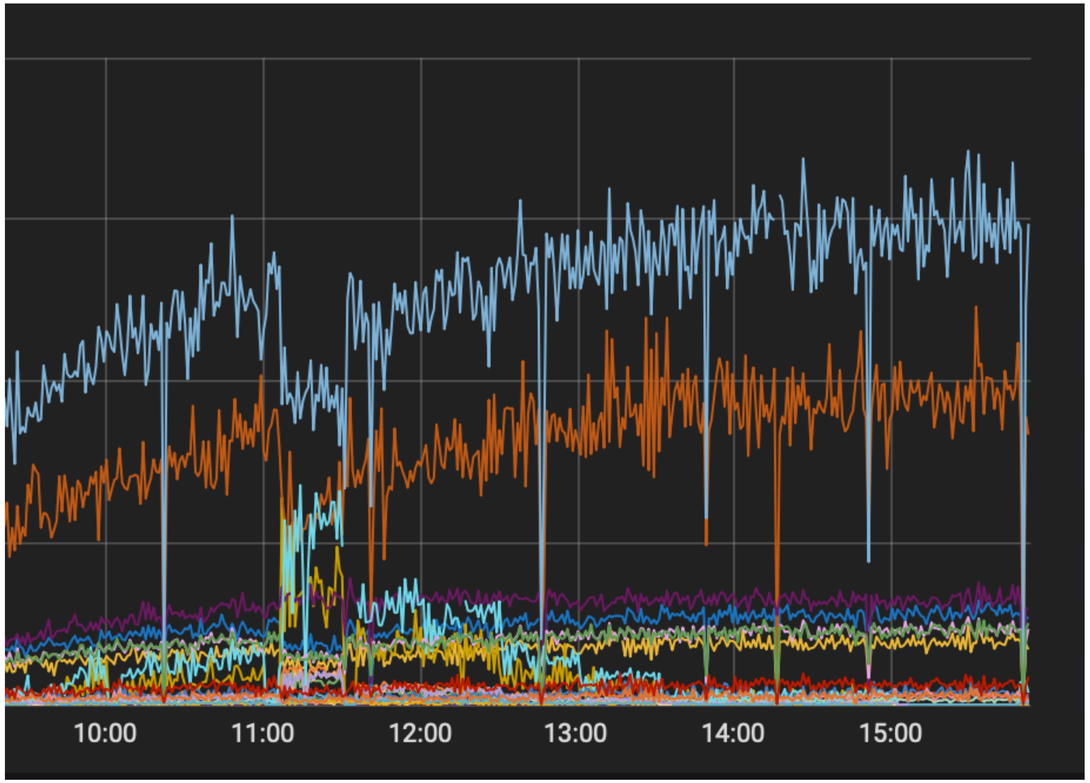
Challenges of Root Cause Localization in Microservice Systems

3. Various types and huge volumes of monitoring data



Background

Metrics or Logs Based Localization Approaches



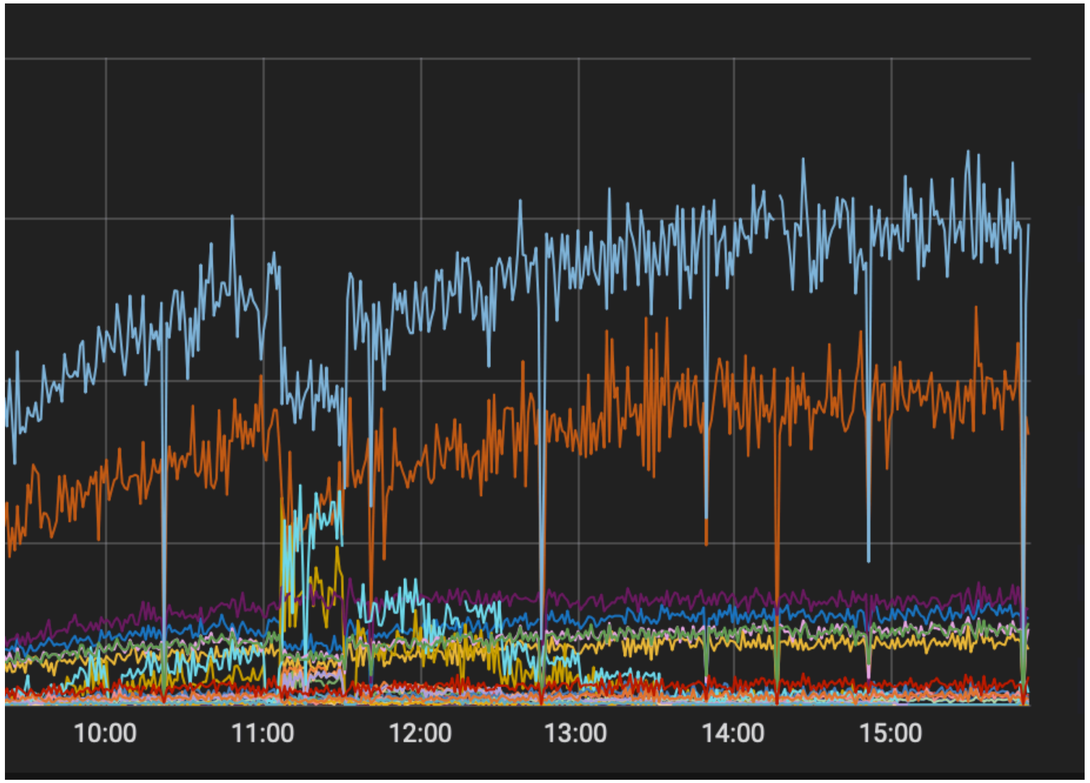
Metrics lack detailed contextual information and fine-grained dependencies of invocations

```
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-15 as the leader reported an err
or: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Retrying leaderEpoch request for partition logs-0 as the leader reported an error: NOT_LEADER_
FOR_PARTITION
[kafka.log][INFO] Opening socket connection to server kafka-zookeeper/10.47.244.48:2181. Will not attempt to aut
henticate using SASL (unknown error)
[kafka.log][INFO] Opening socket connection to server kafka-zookeeper/10.47.244.48:2181. Will not attempt to aut
henticate using SASL (unknown error)
[kafka.log][INFO] Opening socket connection to server kafka-zookeeper/10.47.244.48:2181. Will not attempt to aut
henticate using SASL (unknown error)
[kafka.log][INFO] Opening socket connection to server kafka-zookeeper/10.47.244.48:2181. Will not attempt to aut
henticate using SASL (unknown error)
[kafka.log][INFO] Error sending fetch request (sessionId=839052068, epoch=517118) to node 2: java.nio.channels.C
losedSelectorException.
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-47 as the leader reported an err
or: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-11 as the leader reported an err
or: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-41 as the leader reported an err
or: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-5 as the leader reported an erro
r: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-35 as the leader reported an err
or: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-17 as the leader reported an err
or: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Error sending fetch request (sessionId=1303574239, epoch=127483) to node 0: java.nio.channels.
```

Logs are usually generated in an arbitrary manner and most log messages contain little information for root cause localization

Background

Metrics or Logs Based Localization Approaches



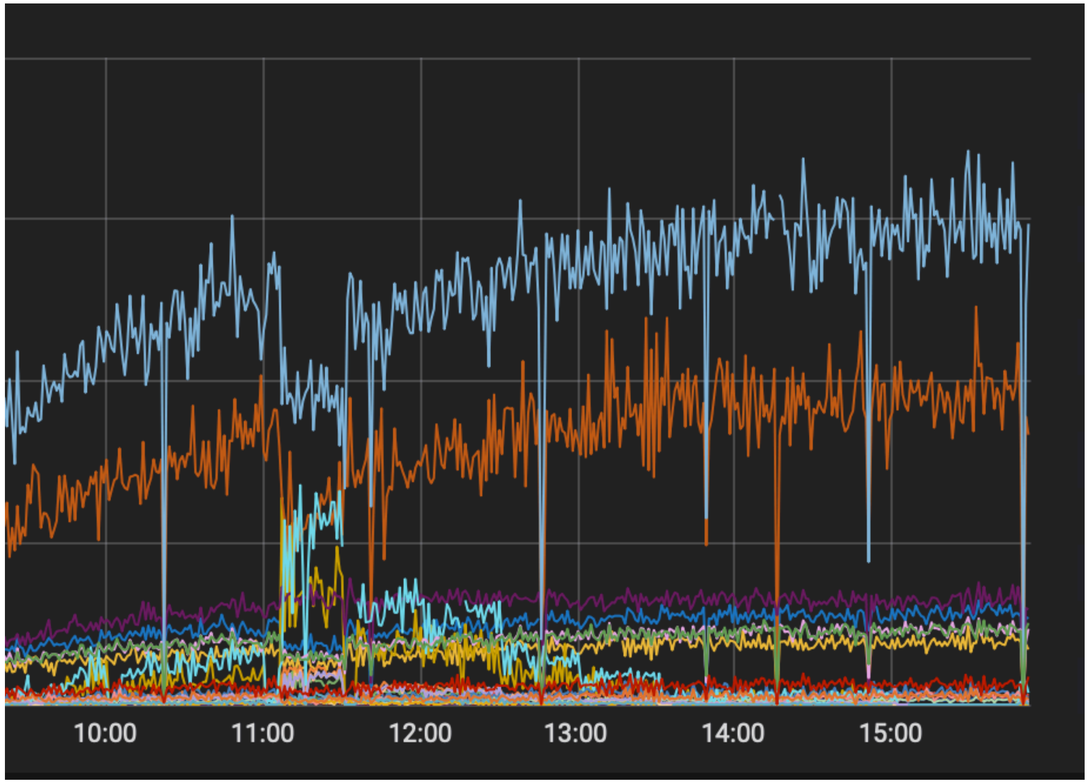
Metrics lack detailed contextual information and fine-grained dependencies of invocations

```
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-15 as the leader reported an err
or: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Retrying leaderEpoch request for partition logs-0 as the leader reported an error: NOT_LEADER_
FOR_PARTITION
[kafka.log][INFO] Opening socket connection to server kafka-zookeeper/10.47.244.48:2181. Will not attempt to aut
henticate using SASL (unknown error)
[kafka.log][INFO] Opening socket connection to server kafka-zookeeper/10.47.244.48:2181. Will not attempt to aut
henticate using SASL (unknown error)
[kafka.log][INFO] Opening socket connection to server kafka-zookeeper/10.47.244.48:2181. Will not attempt to aut
henticate using SASL (unknown error)
[kafka.log][INFO] Opening socket connection to server kafka-zookeeper/10.47.244.48:2181. Will not attempt to aut
henticate using SASL (unknown error)
[kafka.log][INFO] Error sending fetch request (sessionId=839052068, epoch=517118) to node 2: java.nio.channels.C
losedSelectorException.
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-47 as the leader reported an err
or: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-11 as the leader reported an err
or: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-41 as the leader reported an err
or: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-5 as the leader reported an erro
r: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-35 as the leader reported an err
or: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-17 as the leader reported an err
or: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Error sending fetch request (sessionId=1303574239, epoch=127483) to node 0: java.nio.channels.
```

Logs are usually generated in an arbitrary manner and most log messages contain little information for root cause localization

Background

Metrics or Logs Based Localization Approaches



Metrics lack detailed contextual information and fine-grained dependencies of invocations

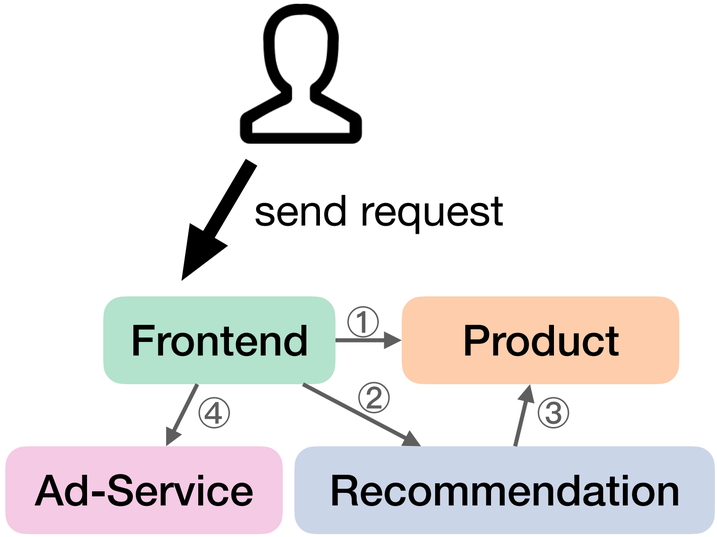
```
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-15 as the leader reported an err
or: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Retrying leaderEpoch request for partition logs-0 as the leader reported an error: NOT_LEADER_
FOR_PARTITION
[kafka.log][INFO] Opening socket connection to server kafka-zookeeper/10.47.244.48:2181. Will not attempt to aut
henticate using SASL (unknown error)
[kafka.log][INFO] Opening socket connection to server kafka-zookeeper/10.47.244.48:2181. Will not attempt to aut
henticate using SASL (unknown error)
[kafka.log][INFO] Opening socket connection to server kafka-zookeeper/10.47.244.48:2181. Will not attempt to aut
henticate using SASL (unknown error)
[kafka.log][INFO] Opening socket connection to server kafka-zookeeper/10.47.244.48:2181. Will not attempt to aut
henticate using SASL (unknown error)
[kafka.log][INFO] Error sending fetch request (sessionId=839052068, epoch=517118) to node 2: java.nio.channels.C
losedSelectorException.
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-47 as the leader reported an err
or: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-11 as the leader reported an err
or: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-41 as the leader reported an err
or: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-5 as the leader reported an erro
r: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-35 as the leader reported an err
or: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Retrying leaderEpoch request for partition __consumer_offsets-17 as the leader reported an err
or: NOT_LEADER_FOR_PARTITION
[kafka.log][INFO] Error sending fetch request (sessionId=1303574239, epoch=127483) to node 0: java.nio.channels.
```

Logs are usually generated in an arbitrary manner and most log messages contain little information for root cause localization

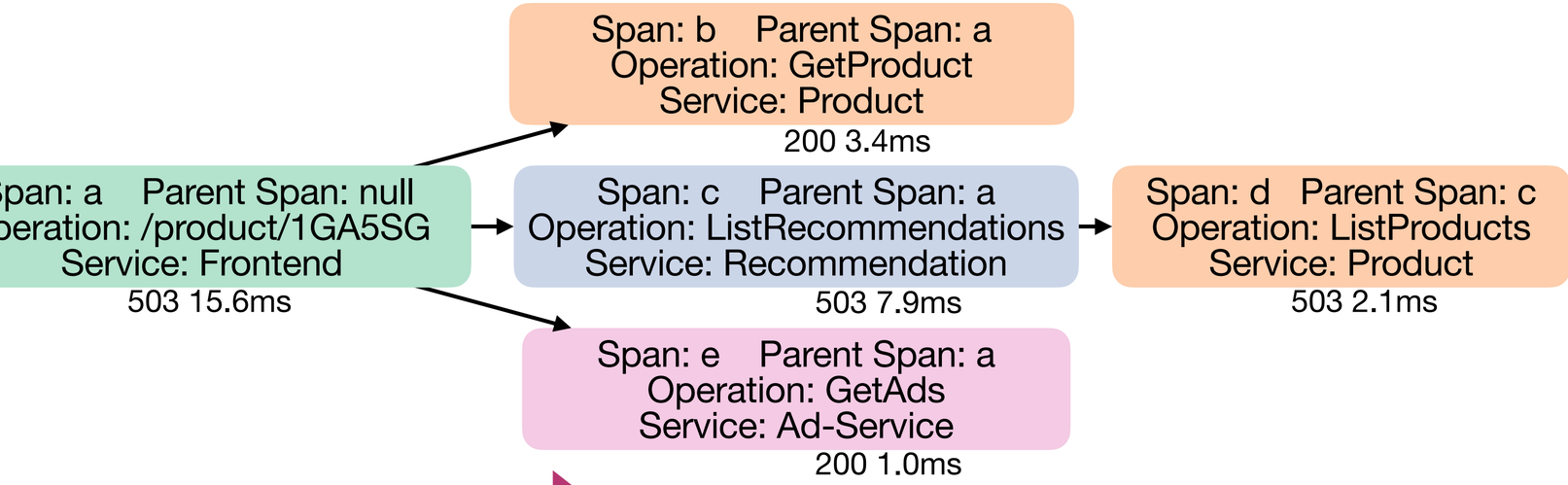
Motivation

Trace-Based Localization Approaches

A trace is the whole execution process invoked by a user request on services



Service Dependency Graph

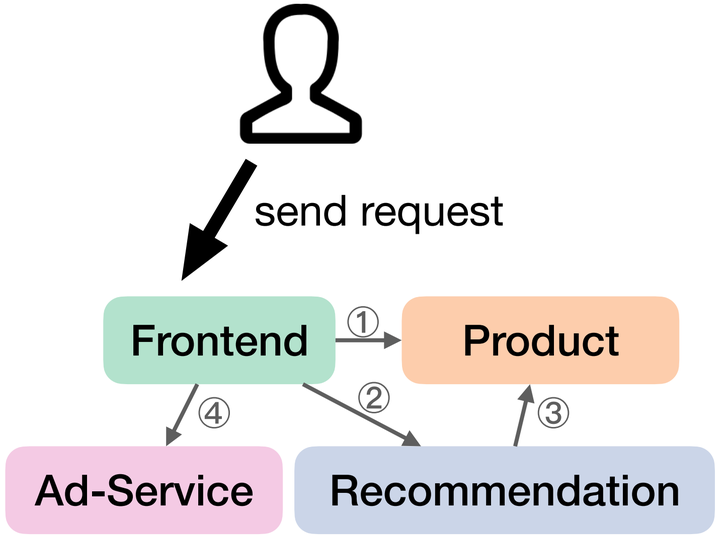


Trace

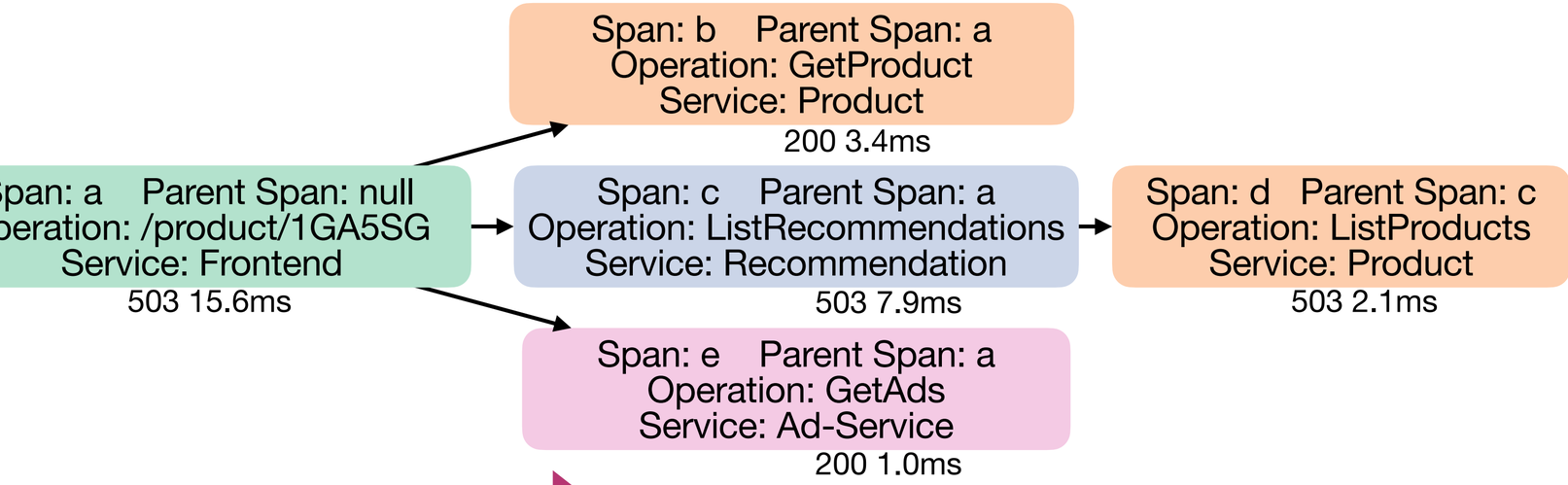
Motivation

Trace-Based Localization Approaches

A trace is the whole execution process invoked by a user request on services



Service Dependency Graph

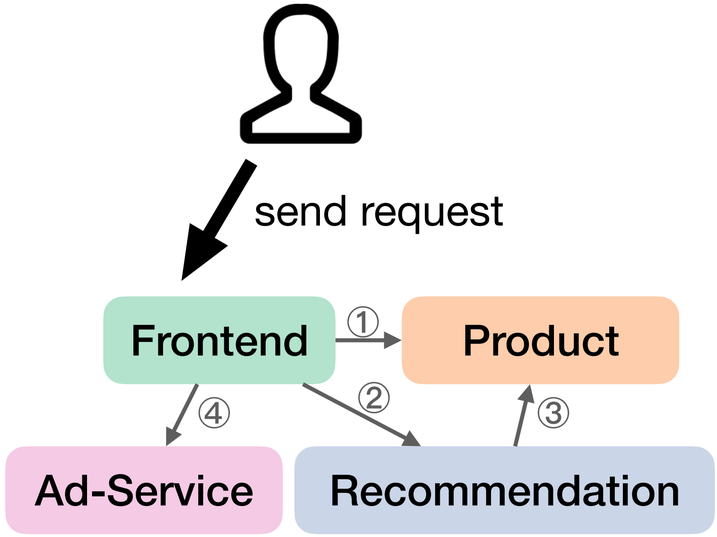


Trace

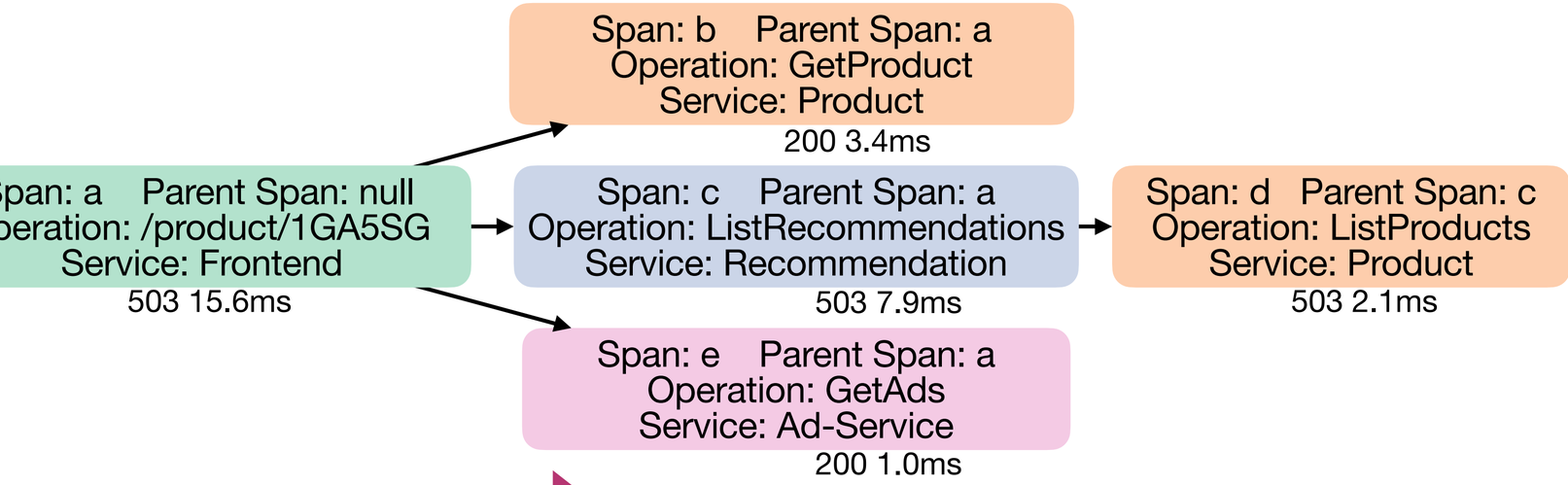
Motivation

Trace-Based Localization Approaches

A trace is the whole execution process invoked by a user request on services



Service Dependency Graph

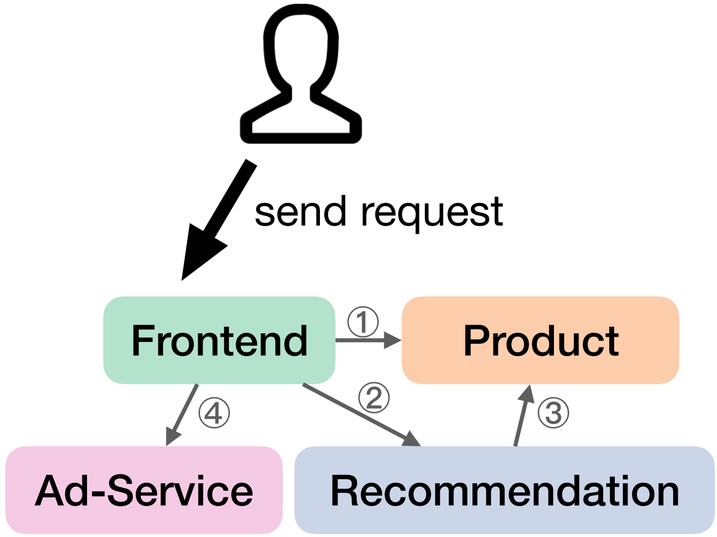


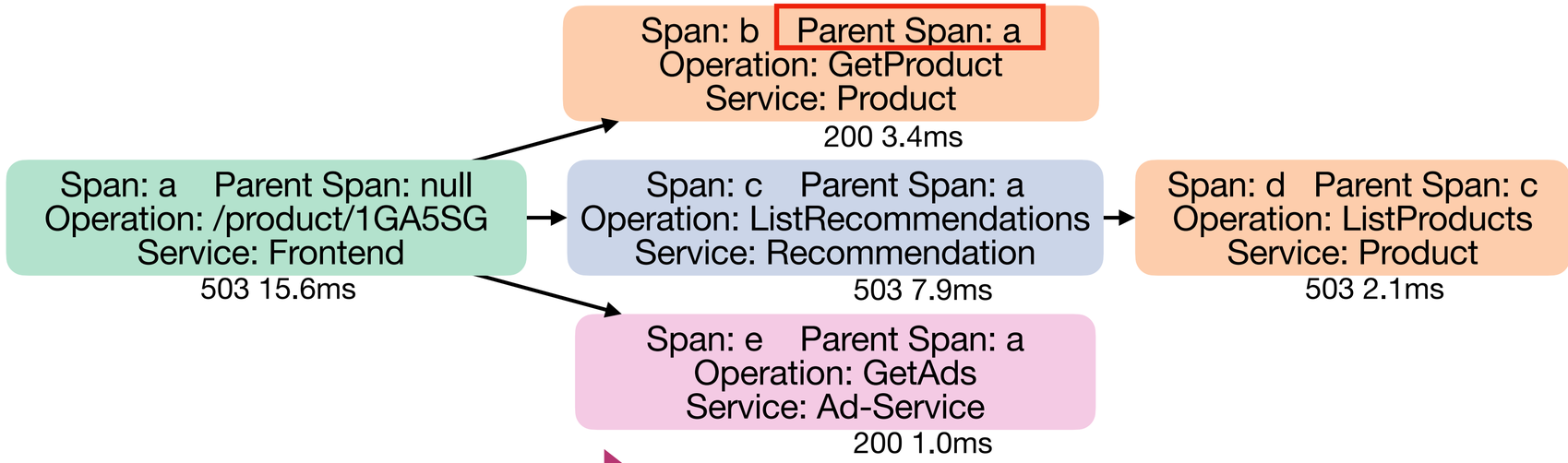
Trace

Motivation

Trace-Based Localization Approaches

A trace is the whole execution process invoked by a user request on services



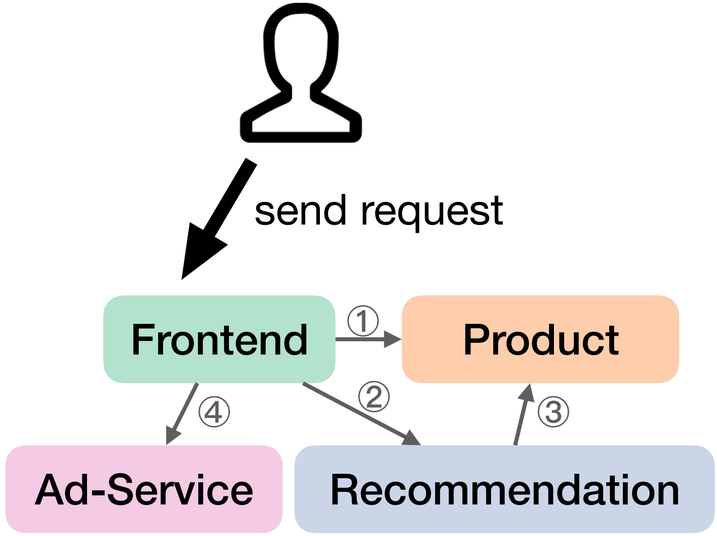


Service Dependency Graph → Trace

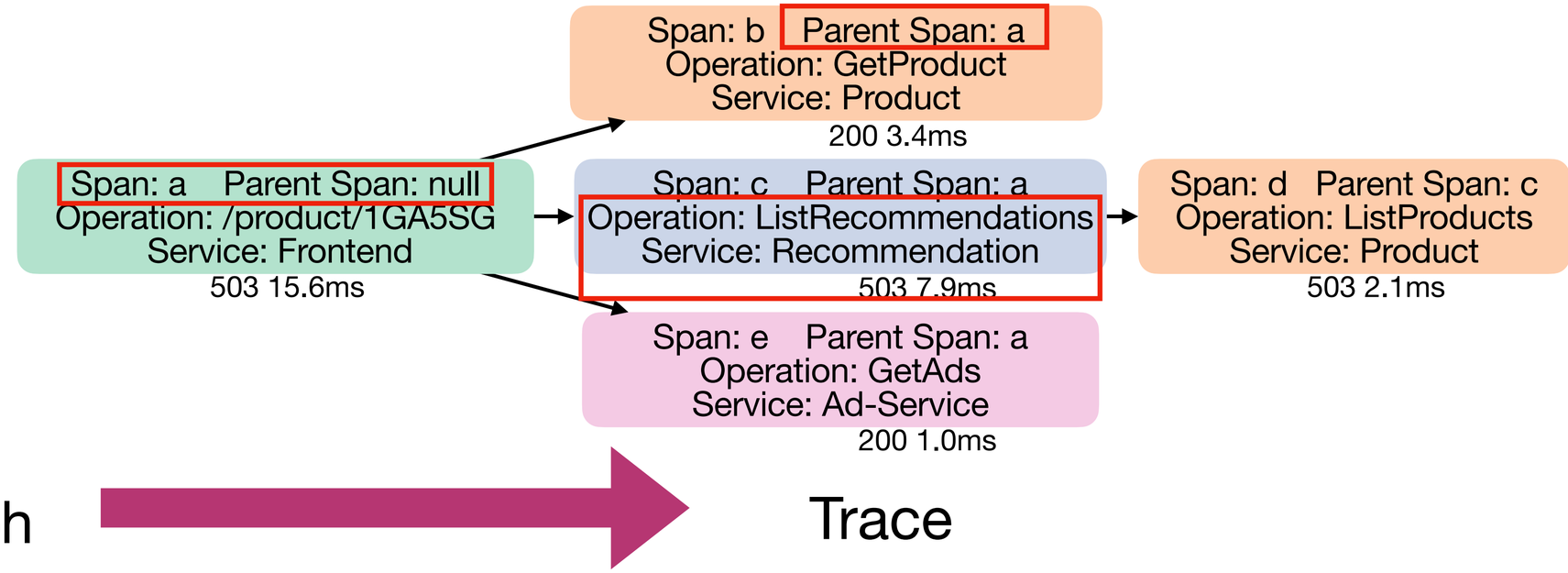
Motivation

Trace-Based Localization Approaches

A trace is the whole execution process invoked by a user request on services



Service Dependency Graph

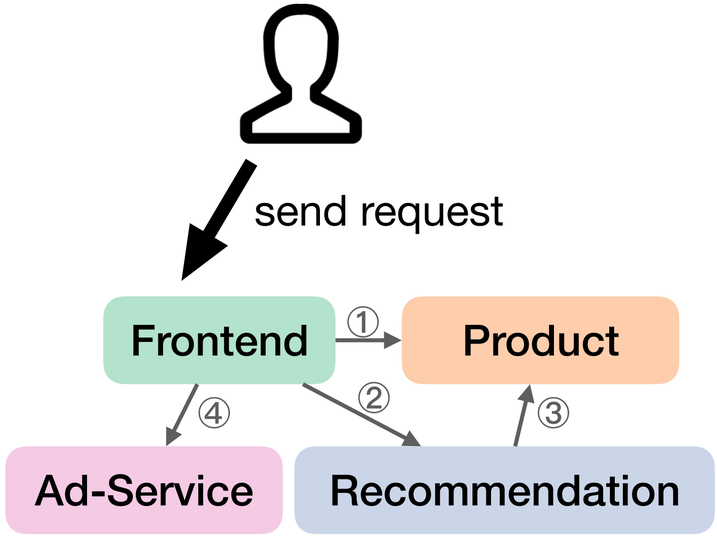


Trace

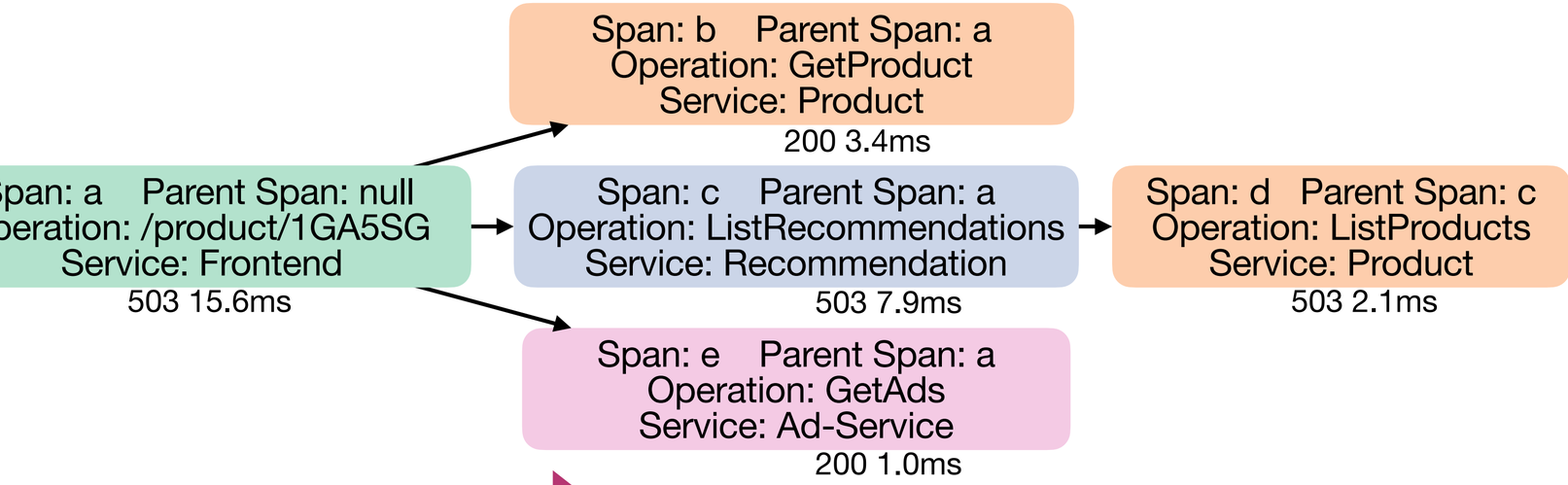
Motivation

Trace-Based Localization Approaches

A trace is the whole execution process invoked by a user request on services



Service Dependency Graph

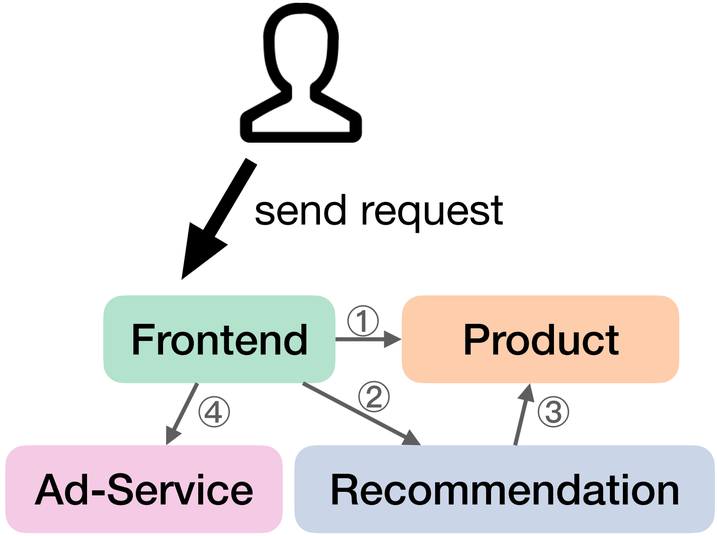


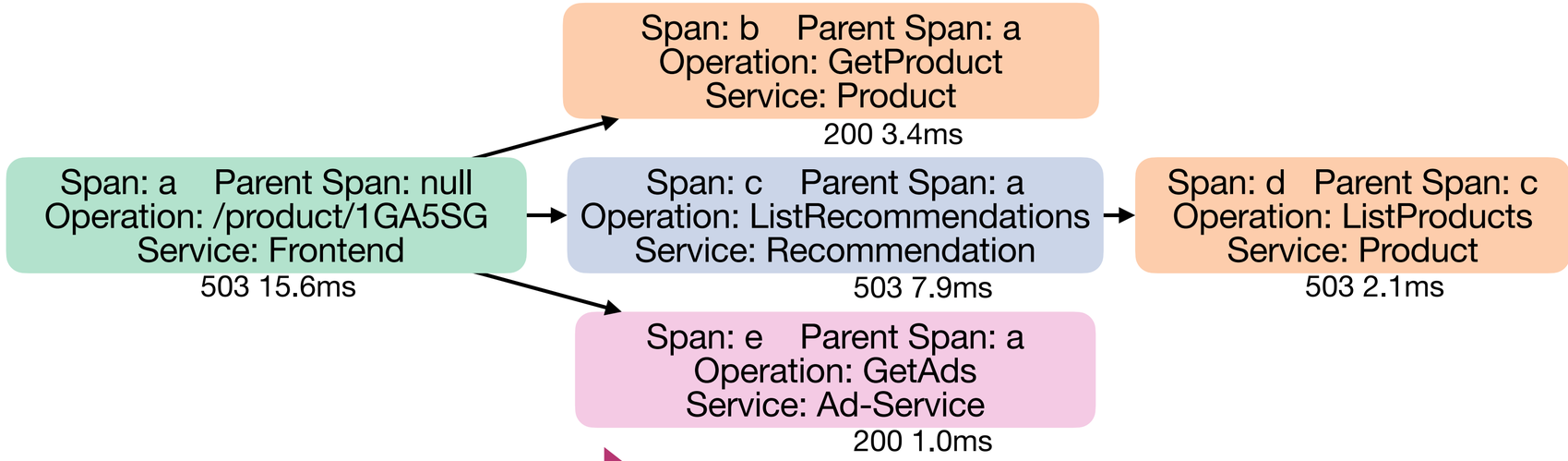
Trace

Motivation

Trace-Based Localization Approaches

A trace is the whole execution process invoked by a user request on services





Service Dependency Graph → Trace

Traces record **detailed dependencies** and enough **contextual information** for root-cause service localization

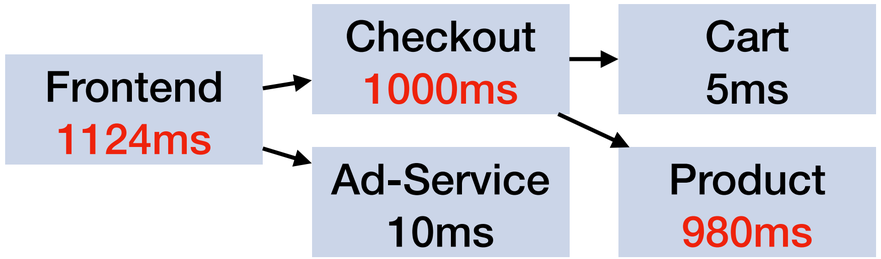
Motivation

Existing Trace-Based Localization Approaches

Motivation

Existing Trace-Based Localization Approaches

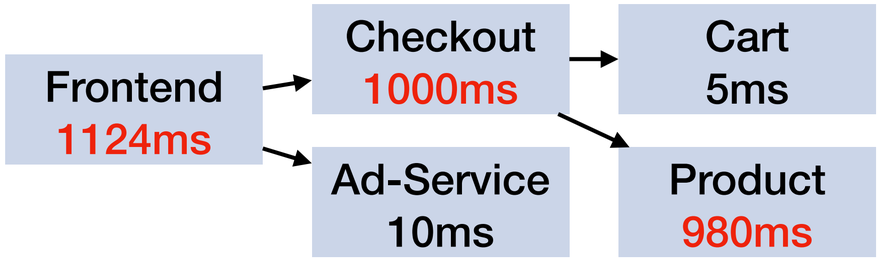
- MicroScope[ICSOC'18] and TraceAnomaly[ISSRE'20] take the most upstream abnormal invocation of an abnormal trace as the root cause.



Motivation

Existing Trace-Based Localization Approaches

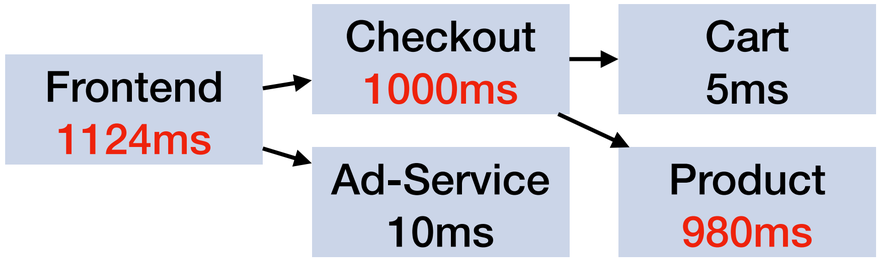
- MicroScope[ICSOC'18] and TraceAnomaly[ISSRE'20] take the most upstream abnormal invocation of an abnormal trace as the root cause.

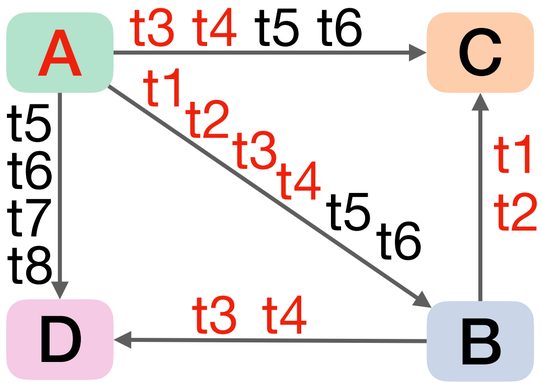


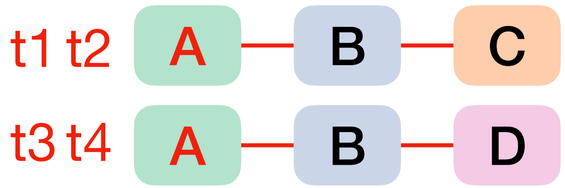
Motivation

Existing Trace-Based Localization Approaches

- MicroScope[ICSOC'18] and TraceAnomaly[ISSRE'20] take the most upstream abnormal invocation of an abnormal trace as the root cause.





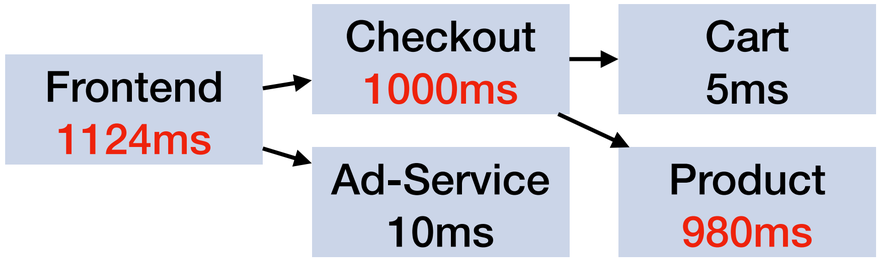


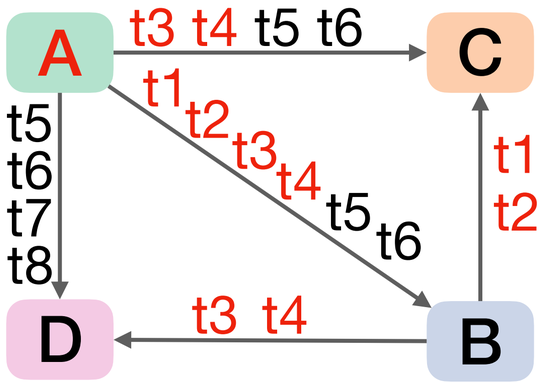
The implementation bug of API /a1 in service A makes further invocations from /a1 abnormal

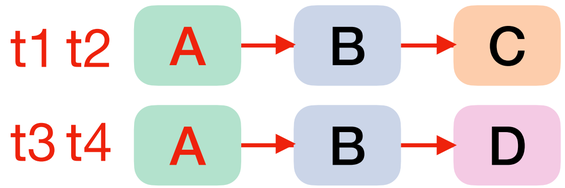
Motivation

Existing Trace-Based Localization Approaches

- MicroScope[ICSOC'18] and TraceAnomaly[ISSRE'20] take the most upstream abnormal invocation of an abnormal trace as the root cause.





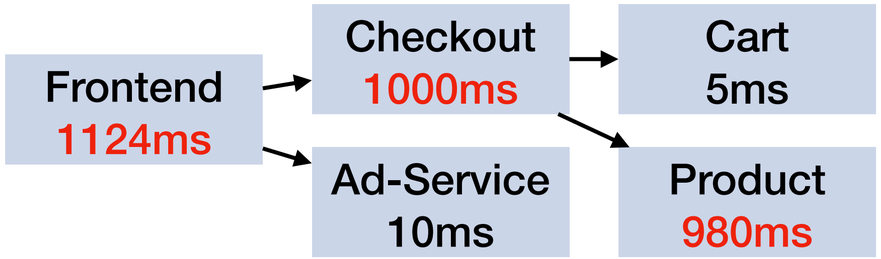


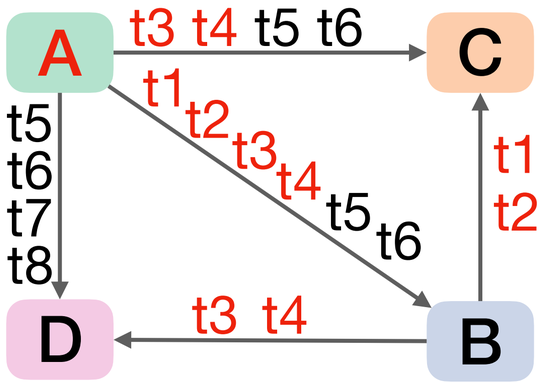
The implementation bug of API /a1 in service A makes further invocations from /a1 abnormal

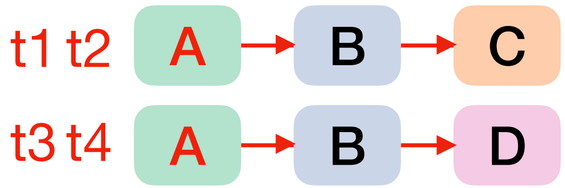
Motivation

Existing Trace-Based Localization Approaches

- MicroScope[ICSOC'18] and TraceAnomaly[ISSRE'20] take the most upstream abnormal invocation of an abnormal trace as the root cause.







The implementation bug of API /a1 in service A makes further invocations from /a1 abnormal

The root-cause service is not the most upstream one

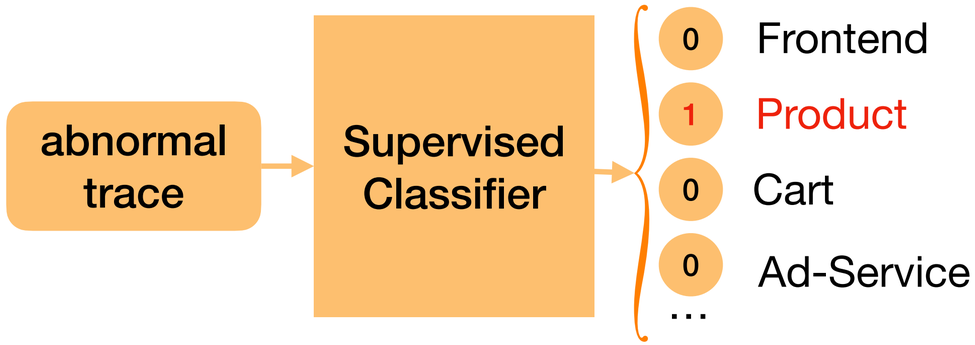
Motivation

Existing Trace-Based Localization Approaches

Motivation

Existing Trace-Based Localization Approaches

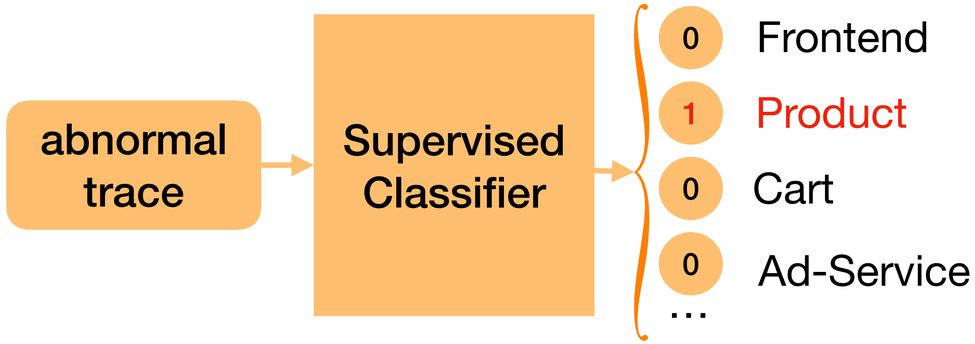
- MEPFL[FSE'19] uses a supervised machine learning model to "classify" the root-cause service for an abnormal trace.

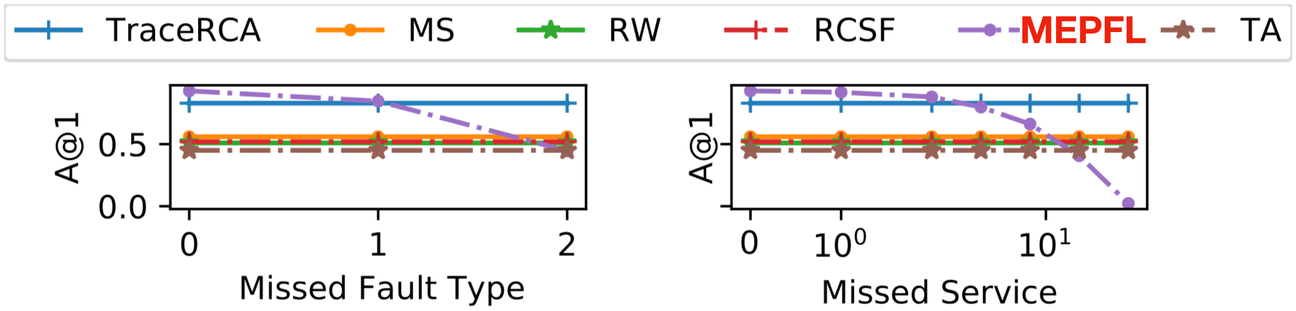


Motivation

Existing Trace-Based Localization Approaches

- MEPFL[FSE'19] uses a supervised machine learning model to "classify" the root-cause service for an abnormal trace.



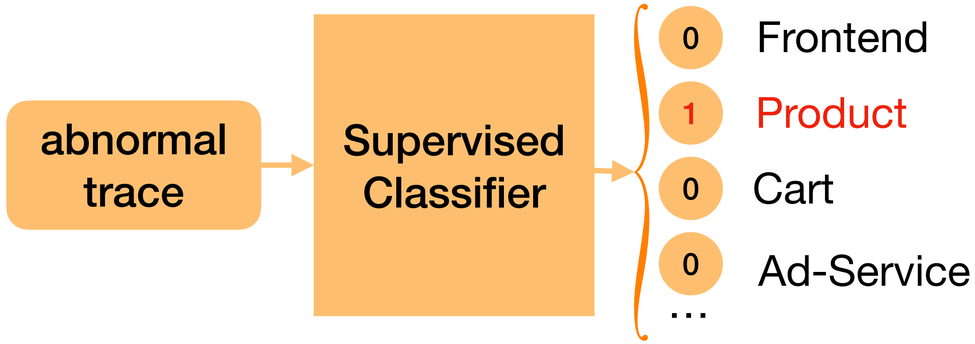


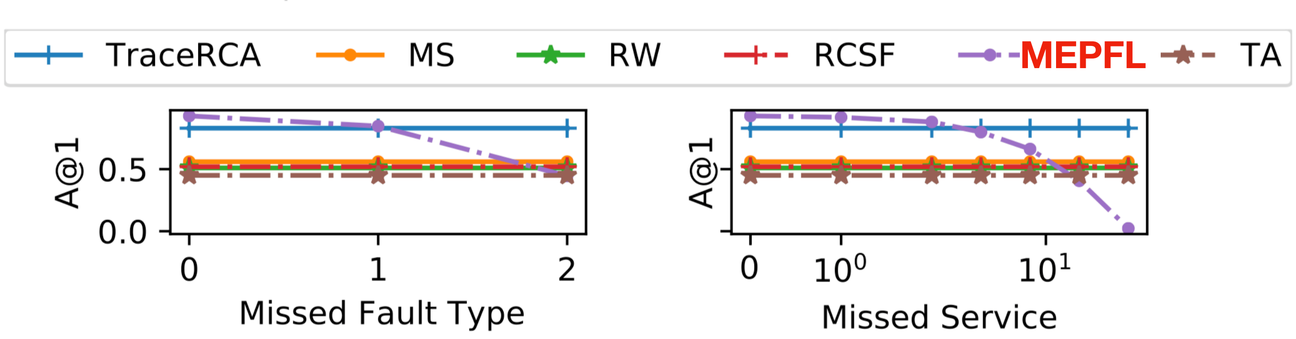
Supervised approaches rely on training data with high coverage on all fault types and microservices

Motivation

Existing Trace-Based Localization Approaches

- MEPFL[FSE'19] uses a supervised machine learning model to "classify" the root-cause service for an abnormal trace.





Supervised approaches rely on training data with high coverage on all fault types and microservices

It is hard for supervised approaches to handle new faults

Motivation

Motivation

The existing trace-based approaches utilize only ***single*** traces and ***abnormal*** traces.

Motivation

The existing trace-based approaches utilize only ***single*** traces and ***abnormal*** traces.

Insight: Intuitively, a microservice covered by more abnormal traces but less normal traces is more likely to be the root cause.

Motivation

The existing trace-based approaches utilize only ***single*** traces and ***abnormal*** traces.

Insight: Intuitively, a microservice covered by more abnormal traces but less normal traces is more likely to be the root cause.

Our insight utilizes the ***statistical information*** based on ***both normal and abnormal traces***

Motivation

Key Metrics

Motivation

Key Metrics

A microservice covered by more abnormal traces but less normal traces is more likely to be the root cause.

Motivation

Key Metrics

A microservice covered by more abnormal traces but less normal traces is more likely to be the root cause.

Motivation

Key Metrics

A microservice covered by more abnormal traces but less normal traces is more likely to be the root cause.

Motivation

Key Metrics

A microservice covered by more abnormal traces but less normal traces is more likely to be the root cause.

$$\text{support}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{abnormal traces}}$$

$$\text{confidence}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{traces passing through } s}$$

Motivation

Key Metrics

A microservice covered by more abnormal traces but less normal traces is more likely to be the root cause.

$$\text{support}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{abnormal traces}}$$

$$\text{confidence}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{traces passing through } s}$$

support $\uparrow$ $\Rightarrow$ “more abnormal traces”

Motivation

Key Metrics

A microservice covered by more abnormal traces but less normal traces is more likely to be the root cause.

$$\text{support}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{abnormal traces}}$$

support ↑ ⇒ “more abnormal traces”

$$\text{confidence}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{traces passing through } s}$$

confidence ↑ ⇒ “less normal traces”

Motivation

Key Metrics

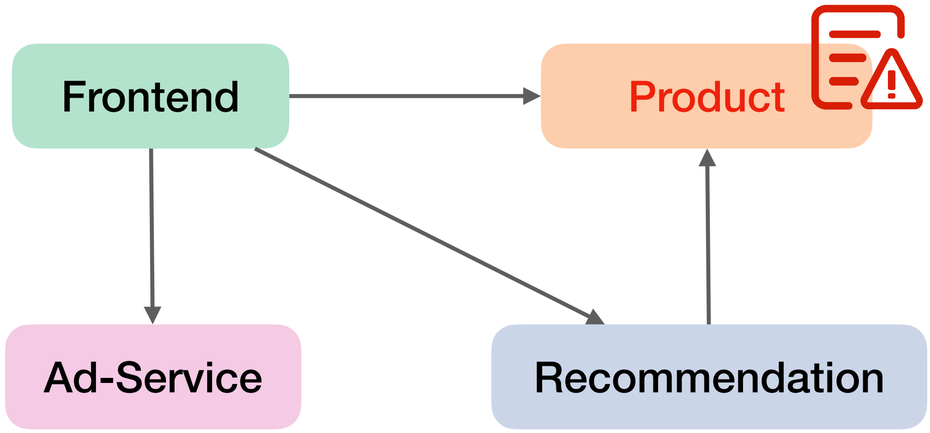
A microservice covered by more abnormal traces but less normal traces is more likely to be the root cause.

$$\text{support}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{abnormal traces}}$$

support ↑ ⇒ “more abnormal traces”

$$\text{confidence}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{traces passing through } s}$$

confidence ↑ ⇒ “less normal traces”



Motivation

Key Metrics

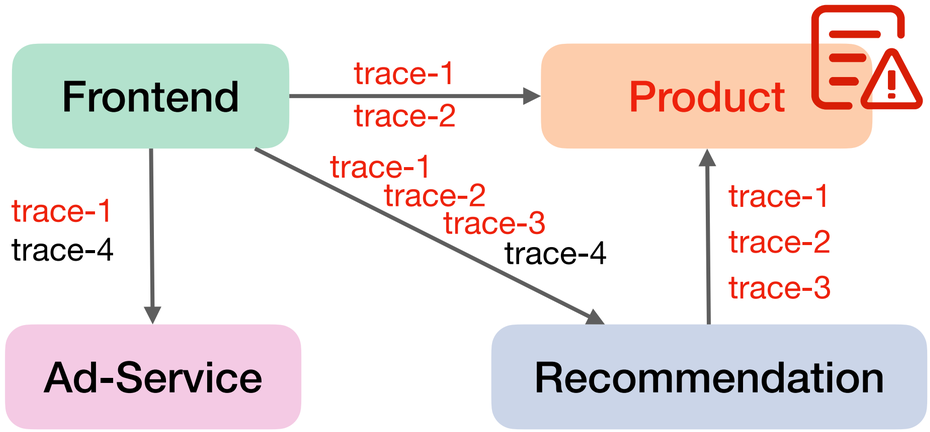
A microservice covered by more abnormal traces but less normal traces is more likely to be the root cause.

$$\text{support}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{abnormal traces}}$$

support ↑ ⇒ “more abnormal traces”

$$\text{confidence}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{traces passing through } s}$$

confidence ↑ ⇒ “less normal traces”



Motivation

Key Metrics

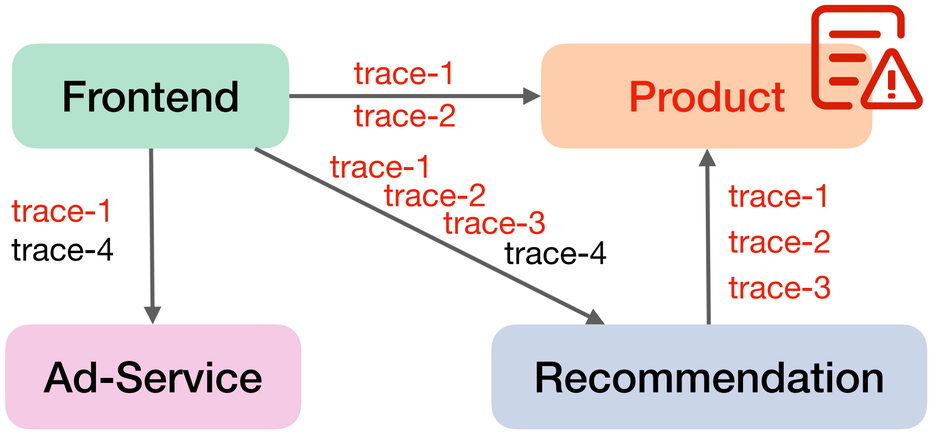
A microservice covered by more abnormal traces but less normal traces is more likely to be the root cause.

$$\text{support}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{abnormal traces}}$$

support ↑ ⇒ “more abnormal traces”

$$\text{confidence}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{traces passing through } s}$$

confidence ↑ ⇒ “less normal traces”



	support	confidence
Frontend		
Product		
Recommendation		
Ad-Service		

Motivation

Key Metrics

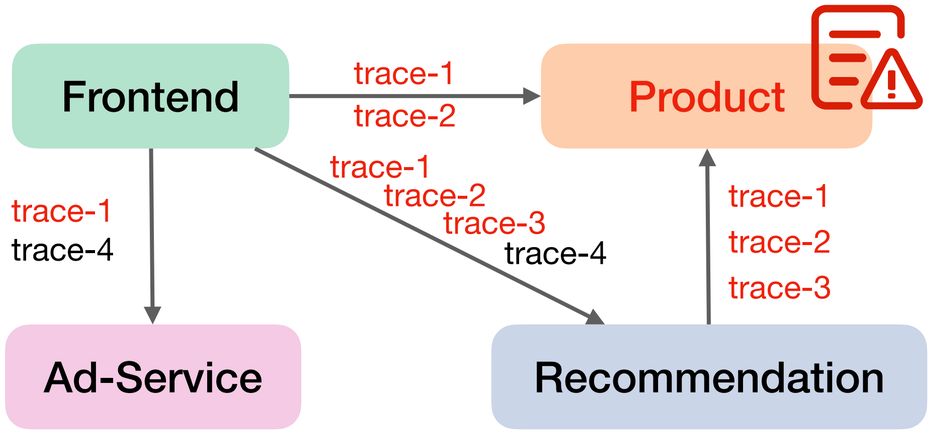
A microservice covered by more abnormal traces but less normal traces is more likely to be the root cause.

$$\text{support}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{abnormal traces}}$$

support ↑ ⇒ “more abnormal traces”

$$\text{confidence}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{traces passing through } s}$$

confidence ↑ ⇒ “less normal traces”



	support	confidence
Frontend		
Product		
Recommendation		
Ad-Service		

Motivation

Key Metrics

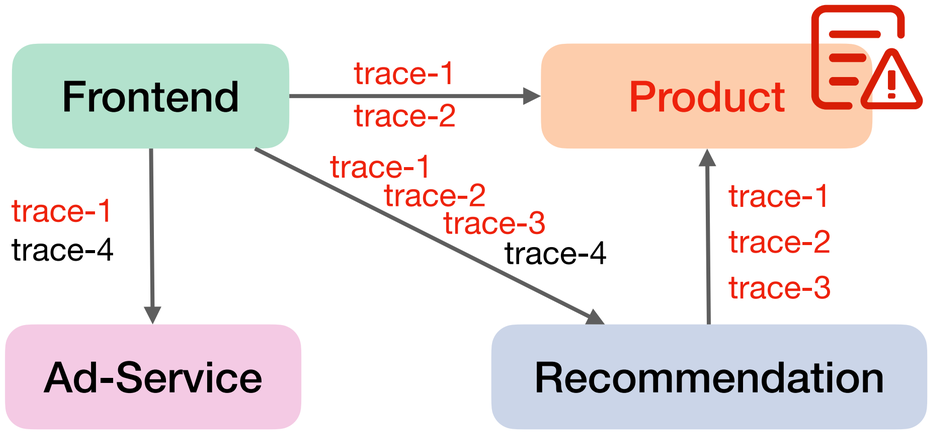
A microservice covered by more abnormal traces but less normal traces is more likely to be the root cause.

$$\text{support}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{abnormal traces}}$$

support ↑ ⇒ “more abnormal traces”

$$\text{confidence}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{traces passing through } s}$$

confidence ↑ ⇒ “less normal traces”



	support	confidence
Frontend	3/3	
Product		
Recommendation		
Ad-Service		

Motivation

Key Metrics

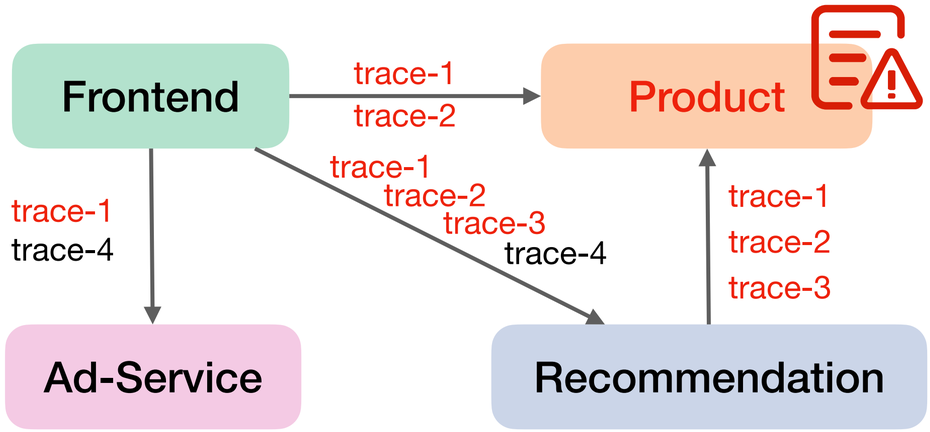
A microservice covered by more abnormal traces but less normal traces is more likely to be the root cause.

$$\text{support}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{abnormal traces}}$$

support ↑ ⇒ “more abnormal traces”

$$\text{confidence}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{traces passing through } s}$$

confidence ↑ ⇒ “less normal traces”



	support	confidence
Frontend	3/3	
Product		
Recommendation		
Ad-Service		

Motivation

Key Metrics

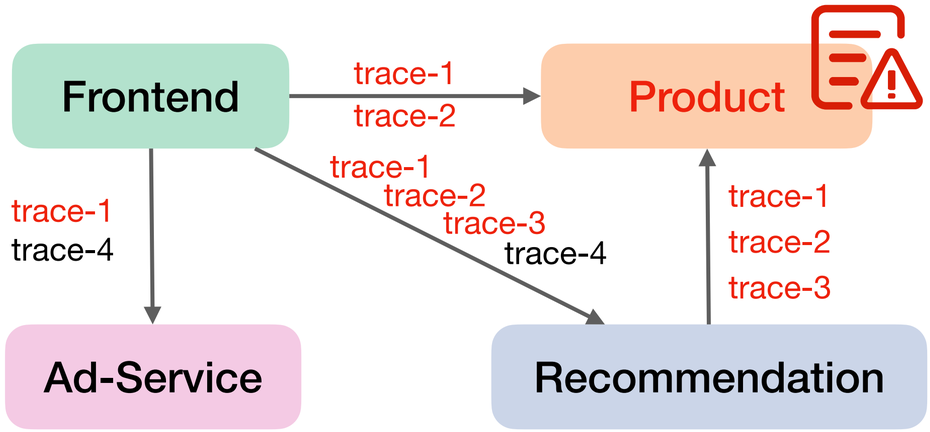
A microservice covered by more abnormal traces but less normal traces is more likely to be the root cause.

$$\text{support}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{abnormal traces}}$$

support ↑ ⇒ “more abnormal traces”

$$\text{confidence}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{traces passing through } s}$$

confidence ↑ ⇒ “less normal traces”



	support	confidence
Frontend	3/3	3/4
Product		
Recommendation		
Ad-Service		

Motivation

Key Metrics

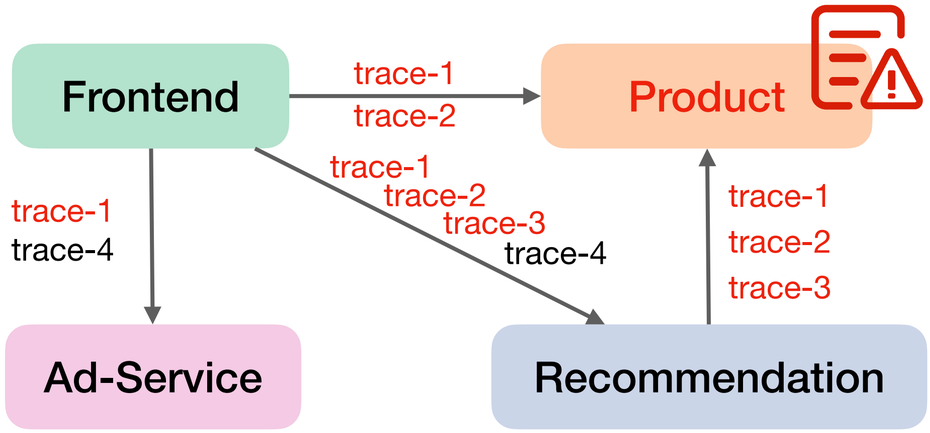
A microservice covered by more abnormal traces but less normal traces is more likely to be the root cause.

$$\text{support}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{abnormal traces}}$$

support ↑ ⇒ “more abnormal traces”

$$\text{confidence}(s) = \frac{\#\text{abnormal traces passing through } s}{\#\text{traces passing through } s}$$

confidence ↑ ⇒ “less normal traces”



	support	confidence
Frontend	3/3	3/4
Product	3/3	3/3
Recommendation	3/3	3/4
Ad-Service	1/3	1/2

Core Idea

1. Multi-Feature Anomaly Detection

Core Idea

1. Multi-Feature Anomaly Detection

We need to detect abnormal traces

Core Idea

1. Multi-Feature Anomaly Detection

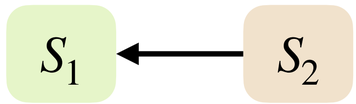
We need to detect abnormal traces

Multi-feature anomaly detection is necessary

Core Idea

1. Multi-Feature Anomaly Detection

We need to detect abnormal traces

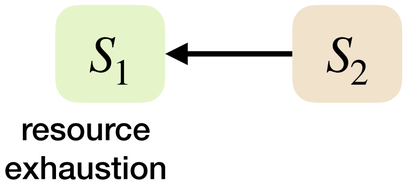


Multi-feature anomaly detection is necessary

Core Idea

1. Multi-Feature Anomaly Detection

We need to detect abnormal traces

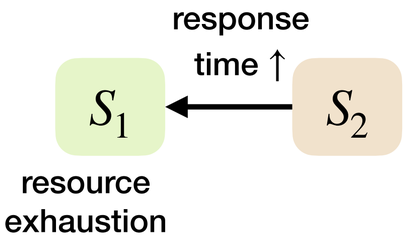


Multi-feature anomaly detection is necessary

Core Idea

1. Multi-Feature Anomaly Detection

We need to detect abnormal traces

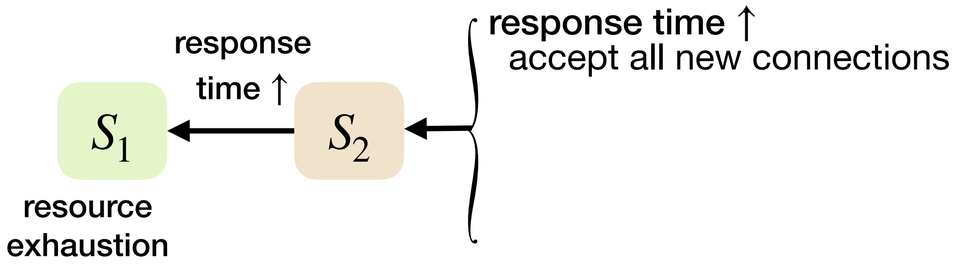


Multi-feature anomaly detection is necessary

Core Idea

1. Multi-Feature Anomaly Detection

We need to detect abnormal traces

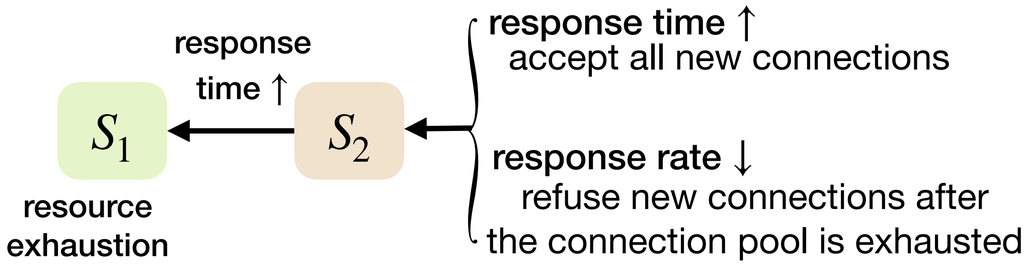


Multi-feature anomaly detection is necessary

Core Idea

1. Multi-Feature Anomaly Detection

We need to detect abnormal traces

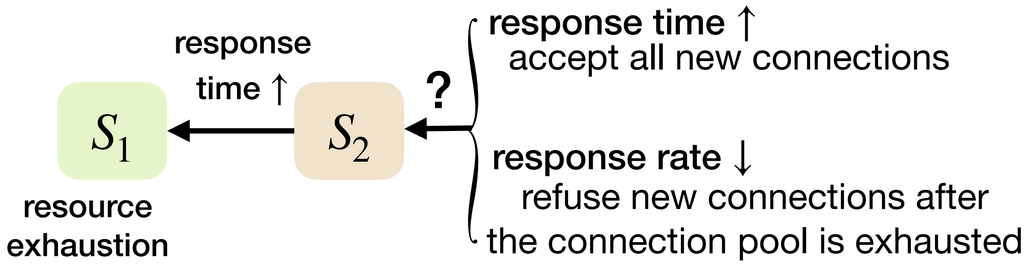


Multi-feature anomaly detection is necessary

Core Idea

1. Multi-Feature Anomaly Detection

We need to detect abnormal traces

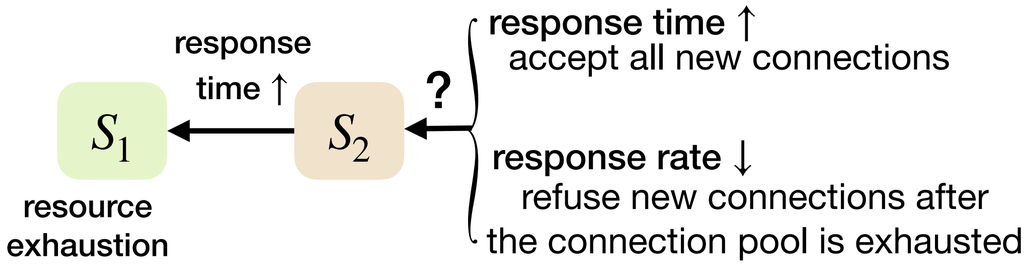


Multi-feature anomaly detection is necessary

Core Idea

1. Multi-Feature Anomaly Detection

We need to detect abnormal traces



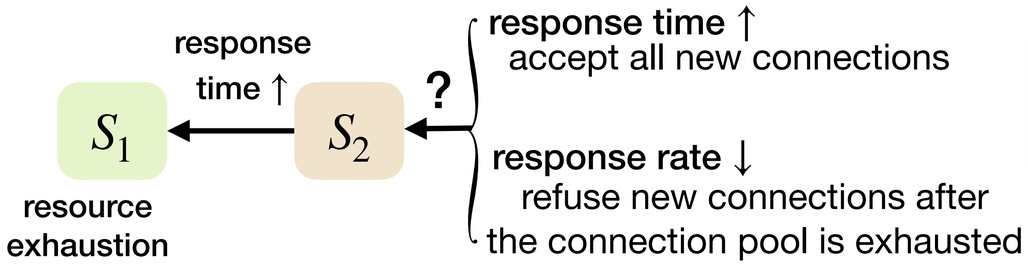
Multi-feature anomaly detection is necessary

Only some features are related to the concerned fault, but there could be also anomalies in other irrelevant features

Core Idea

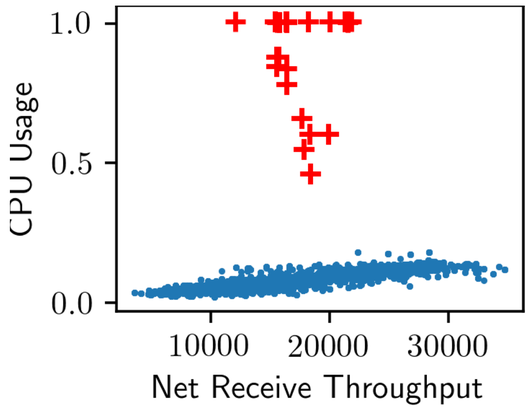
1. Multi-Feature Anomaly Detection

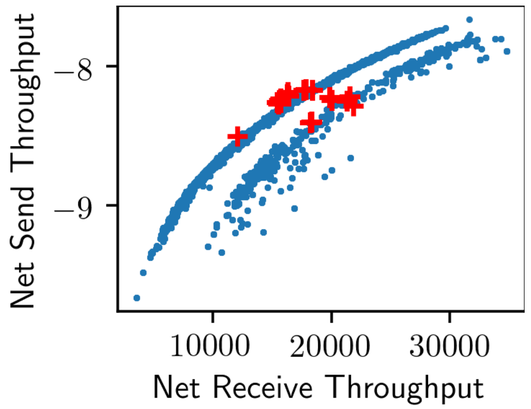
We need to detect abnormal traces



Multi-feature anomaly detection is necessary

Only some features are related to the concerned fault, but there could be also anomalies in other irrelevant features

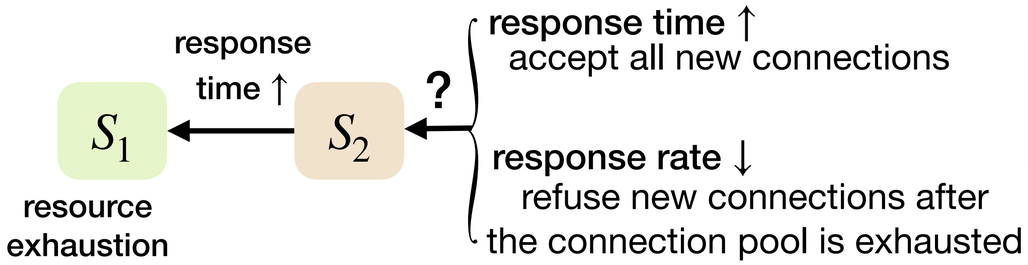




Core Idea

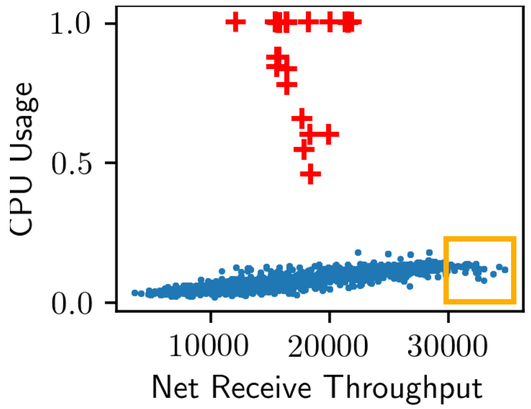
1. Multi-Feature Anomaly Detection

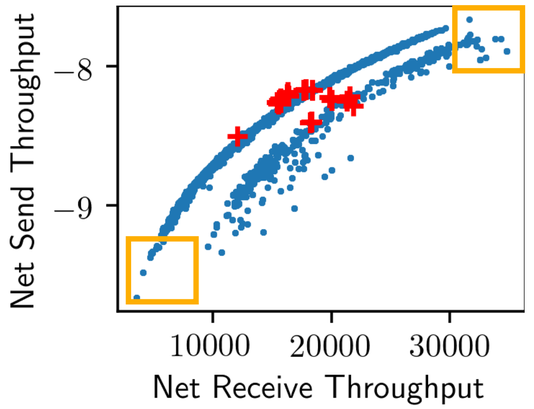
We need to detect abnormal traces



Multi-feature anomaly detection is necessary

Only some features are related to the concerned fault, but there could be also anomalies in other irrelevant features



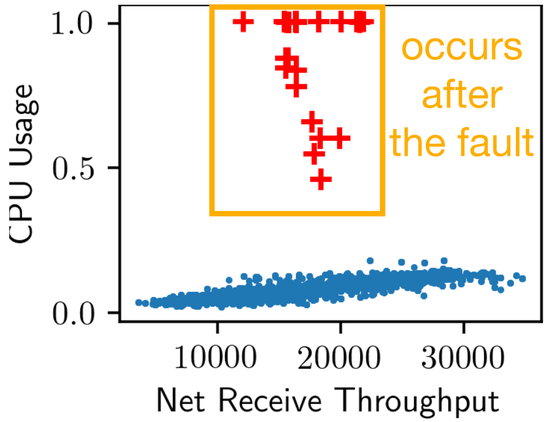


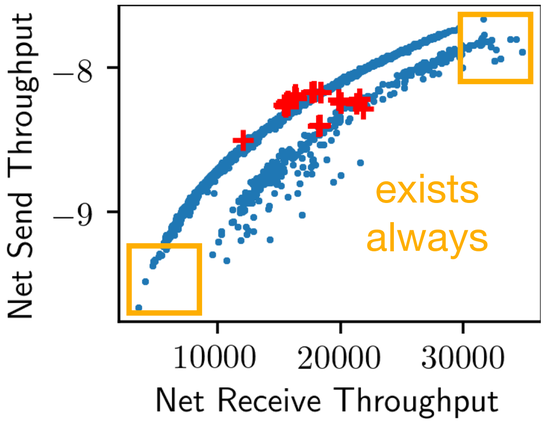
Core Idea

1. Multi-Feature Anomaly Detection

Core Idea

1. Multi-Feature Anomaly Detection

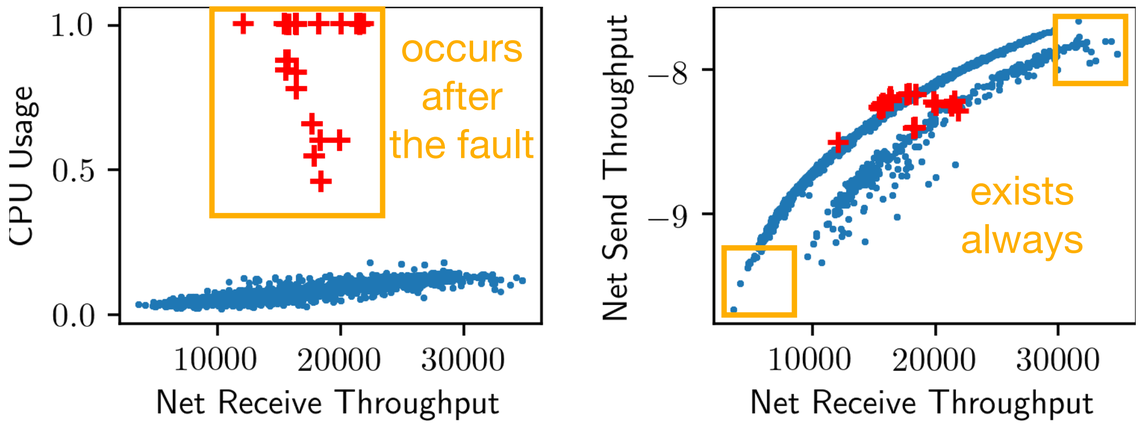




The number of anomalies with respect to useful features and irrelevant features are different

Core Idea

1. Multi-Feature Anomaly Detection

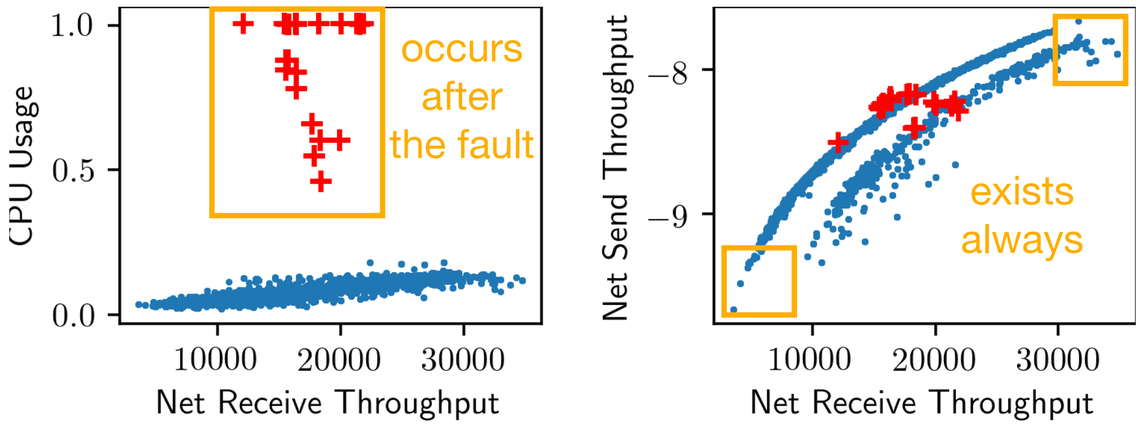


The number of anomalies with respect to useful features and irrelevant features are different

Highlight 1: We select useful features by testing whether the distribution of abnormal and normal invocations with respect to a feature changes after the fault happens

Core Idea

1. Multi-Feature Anomaly Detection



The number of anomalies with respect to useful features and irrelevant features are different

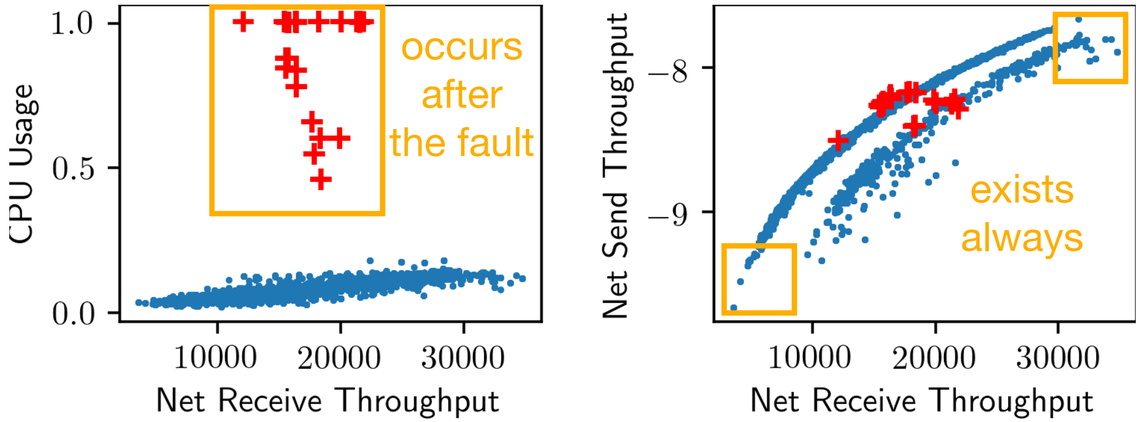
Highlight 1: We select useful features by testing whether the distribution of abnormal and normal invocations with respect to a feature changes after the fault happens

CPU usage	Before	After
Normal	99%	90%
Abnormal	1%	10%

✅ Useful feature

Core Idea

1. Multi-Feature Anomaly Detection



The number of anomalies with respect to useful features and irrelevant features are different

Highlight 1: We select useful features by testing whether the distribution of abnormal and normal invocations with respect to a feature changes after the fault happens

CPU usage	Before	After
Normal	99%	90%
Abnormal	1%	10%

✅ Useful feature

Throughput	Before	After
Normal	98%	98%
Abnormal	2%	2%

❌ Irrelevant feature

Core Idea

2. Mining Microservice Sets

Core Idea

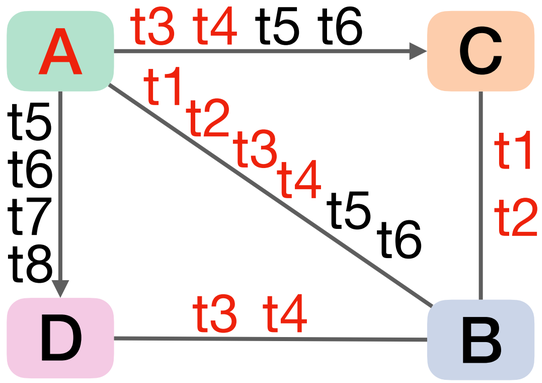
2. Mining Microservice Sets

Highlight 2: (Extending Our Insight)
*We mine **microservice sets** with high supports and confidences*
*rather than **single microservices***

Core Idea

2. Mining Microservice Sets

Highlight 2: (Extending Our Insight)
*We mine **microservice sets** with high supports and confidences rather than **single microservices***

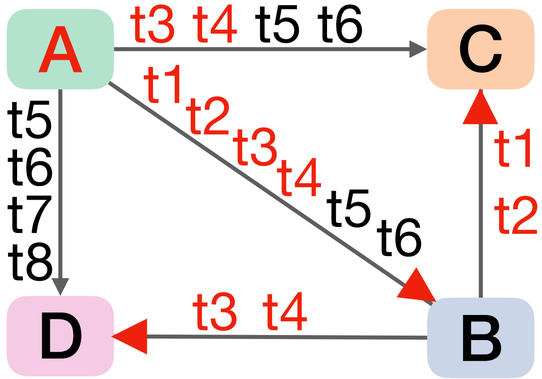


API /a1 of service A is faulty.
Only those traces following pattern A->B->* are abnormal.

Core Idea

2. Mining Microservice Sets

Highlight 2: (Extending Our Insight)
*We mine **microservice sets** with high supports and confidences rather than **single microservices***

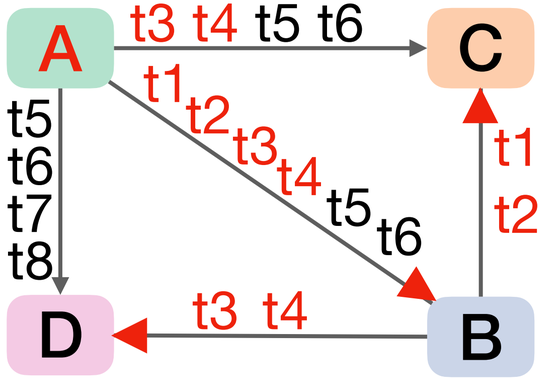


API /a1 of service A is faulty.
Only those traces following pattern A->B->* are abnormal.

Core Idea

2. Mining Microservice Sets

Highlight 2: (Extending Our Insight)
*We mine **microservice sets** with high supports and confidences rather than **single microservices***



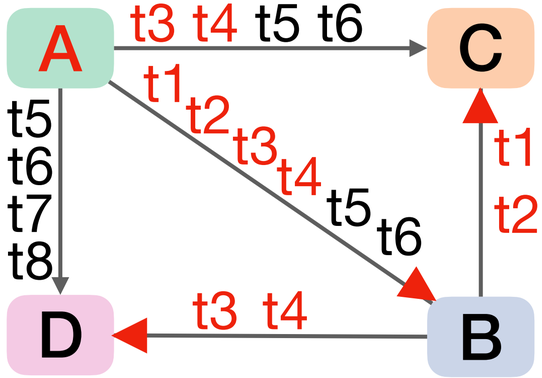
	support	confidence
A	4/4	4/6
B	4/4	4/6
C	4/4	4/6
D	2/4	2/6

API /a1 of service A is faulty.
Only those traces following pattern A->B->* are abnormal.

Core Idea

2. Mining Microservice Sets

Highlight 2: (Extending Our Insight)
*We mine **microservice sets** with high supports and confidences rather than **single microservices***



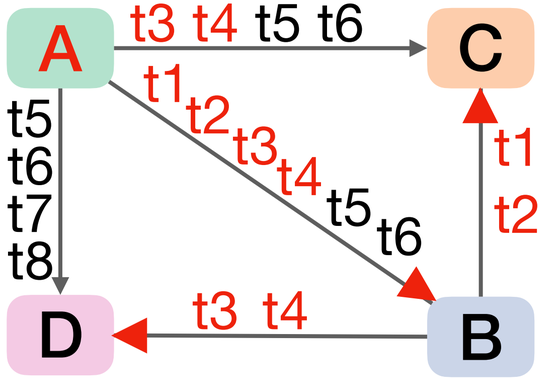
	support	confidence
A	4/4	4/6
B	4/4	4/6
C	4/4	4/6
D	2/4	2/6

API /a1 of service A is faulty.
Only those traces following pattern A->B->* are abnormal.

Core Idea

2. Mining Microservice Sets

Highlight 2: (Extending Our Insight)
*We mine **microservice sets** with high supports and confidences rather than **single microservices***



API /a1 of service A is faulty.
Only those traces following pattern A->B->* are abnormal.

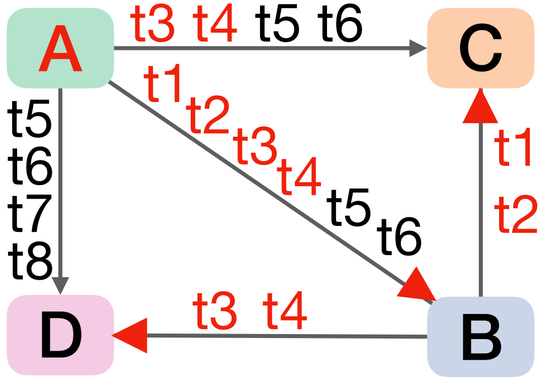
	support	confidence
A	4/4	4/6
B	4/4	4/6
C	4/4	4/6
D	2/4	2/6

	support	confidence
{A, B}	4/4	4/6
{A, C}	2/4	2/4
{B, C}	2/4	2/2
{B, D}	2/4	2/2

Core Idea

2. Mining Microservice Sets

Highlight 2: (Extending Our Insight)
*We mine **microservice sets** with high supports and confidences rather than **single microservices***



API /a1 of service A is faulty.
Only those traces following pattern A->B->* are abnormal.

	support	confidence
A	4/4	4/6
B	4/4	4/6
C	4/4	4/6
D	2/4	2/6

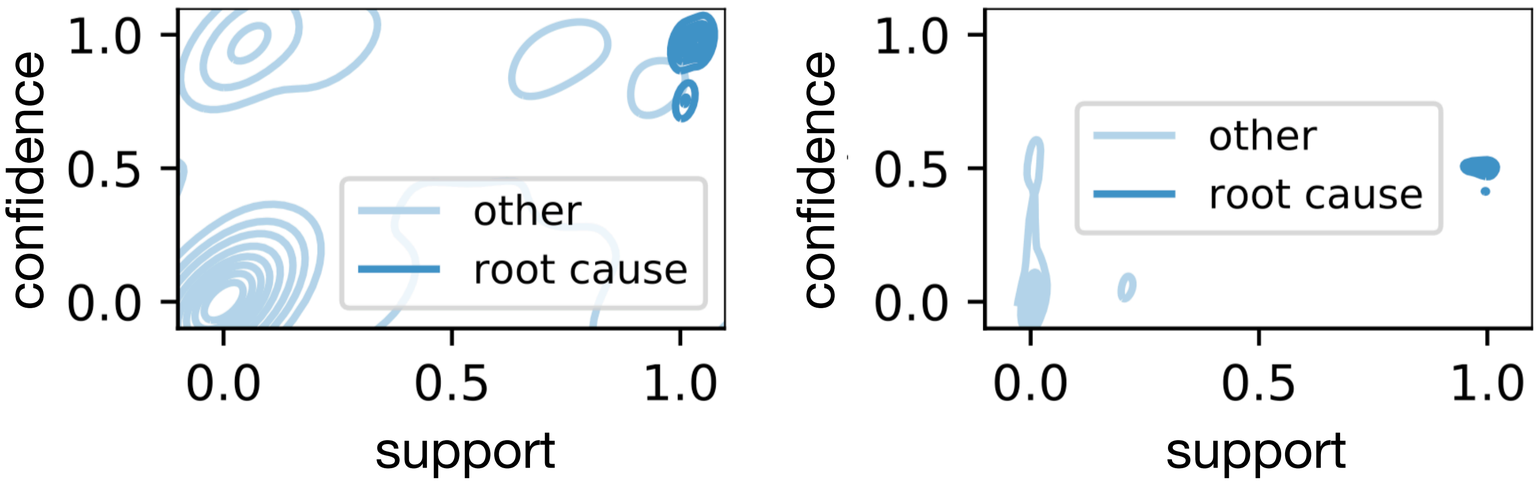
❌

	support	confidence
{A, B}	4/4	4/6
{A, C}	2/4	2/4
{B, C}	2/4	2/2
{B, D}	2/4	2/2

✅

Core Idea

2. Mining Microservice Sets



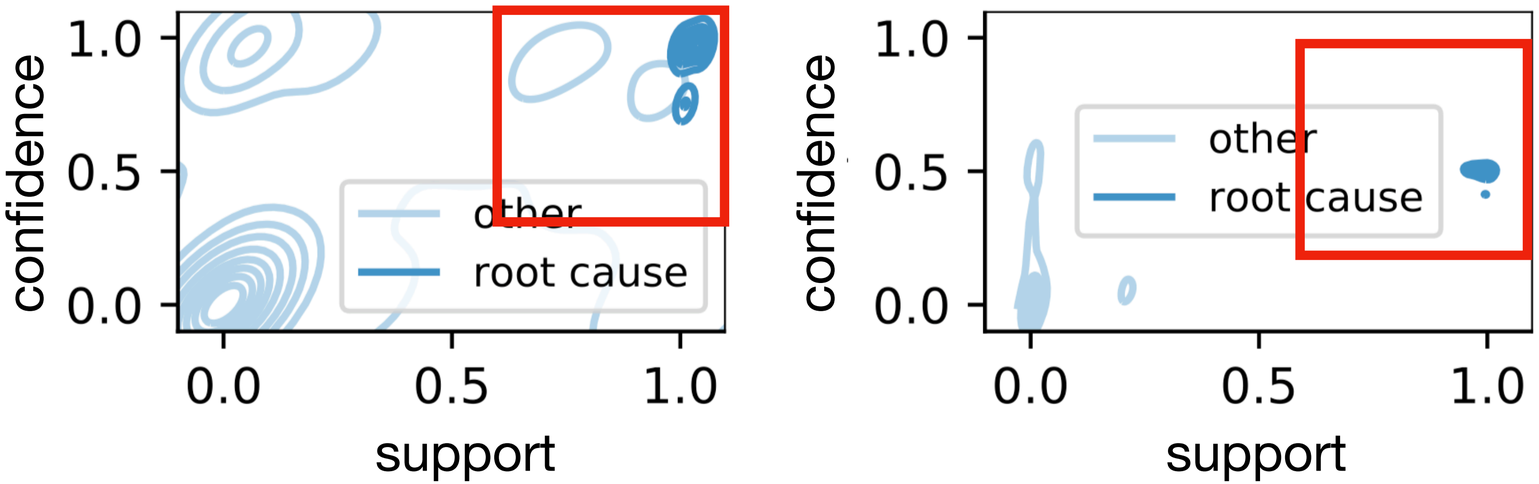
(a) Google Online Boutique (b) Production System $\mathcal{B}$

Fig. 4: The distribution of support and confidence

Our insight and the two key metrics are validated on the two systems

Core Idea

2. Mining Microservice Sets



(a) Google Online Boutique (b) Production System $\mathcal{B}$

Fig. 4: The distribution of support and confidence

Our insight and the two key metrics are validated on the two systems

Core Idea

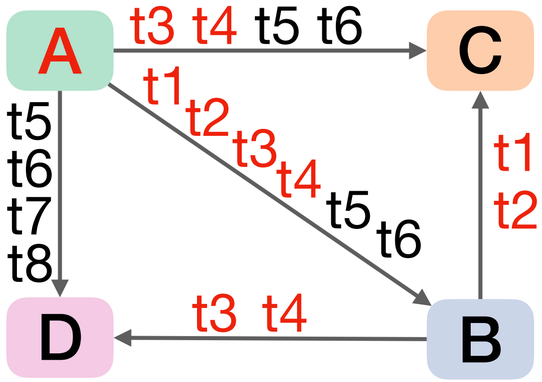
3. Microservice Ranking

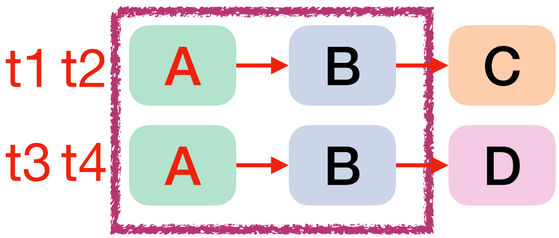
Target Output: A ordered list of microservices

Core Idea

3. Microservice Ranking

Target Output: A ordered list of microservices

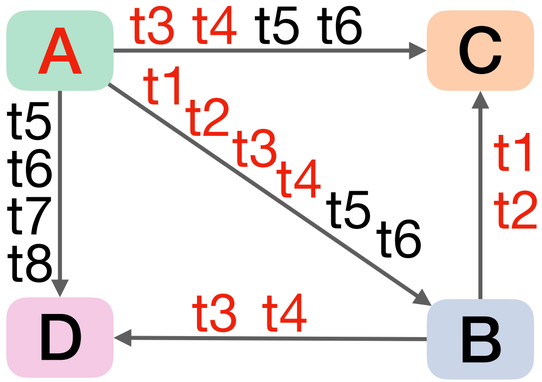


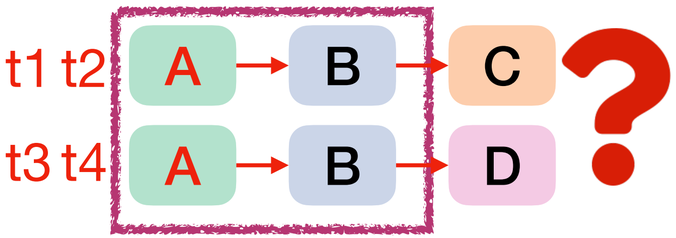


Core Idea

3. Microservice Ranking

Target Output: A ordered list of microservices



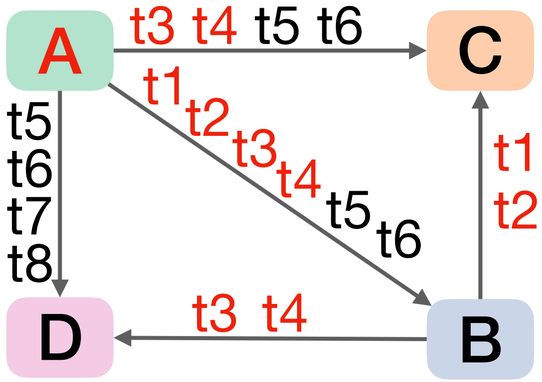


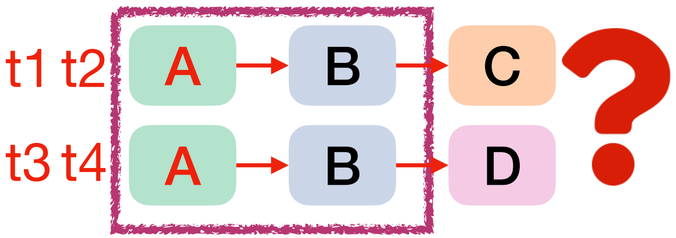
We have mined **{A, B}**, but which one should be further investigated by operators first?

Core Idea

3. Microservice Ranking

Target Output: A ordered list of microservices





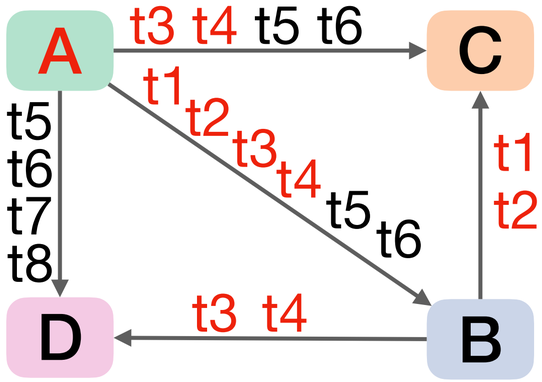
We have mined **{A, B}**, but which one should be further investigated by operators first?

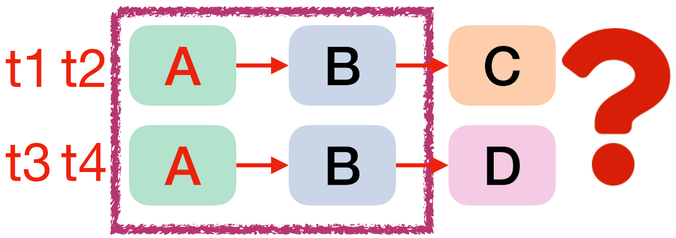
Existing approaches presume the most upstream service is the root cause (page. 10)

Core Idea

3. Microservice Ranking

Target Output: A ordered list of microservices





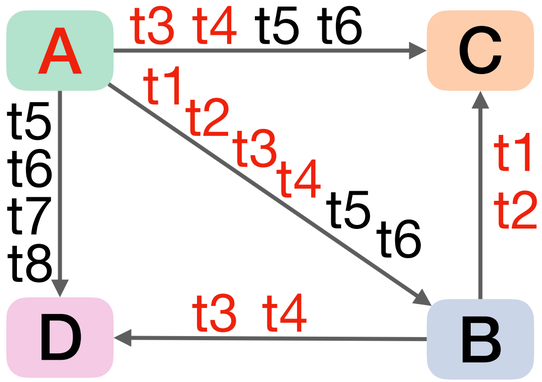
We have mined **{A, B}**, but which one should be further investigated by operators first?

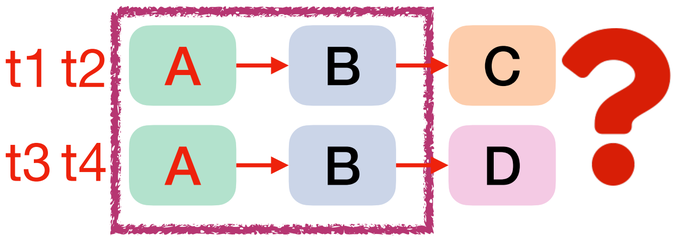
Existing approaches presume the most upstream service is the root cause (page. 10)

Core Idea

3. Microservice Ranking

Target Output: A ordered list of microservices





We have mined **{A, B}**, but which one should be further investigated by operators first?

Existing approaches presume the most upstream service is the root cause (page. 10)

Core Idea

3. Microservice Ranking

Core Idea

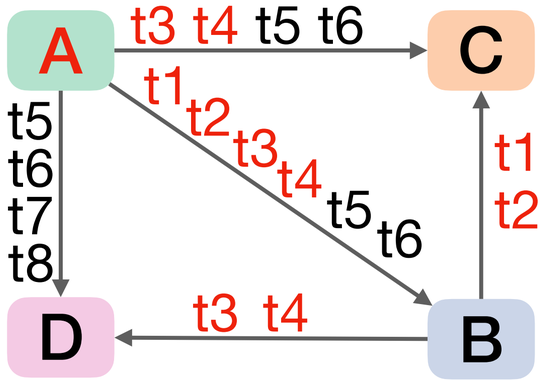
3. Microservice Ranking

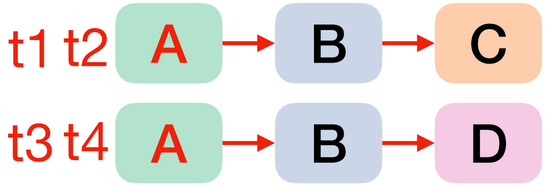
Highlight 2: For any service on an abnormal trace, if it has both in-coming and out-coming abnormal invocations, it is probably just propagating the anomaly.

Core Idea

3. Microservice Ranking

Highlight 2: For any service on an abnormal trace, if it has both in-coming and out-coming abnormal invocations, it is probably just propagating the anomaly.

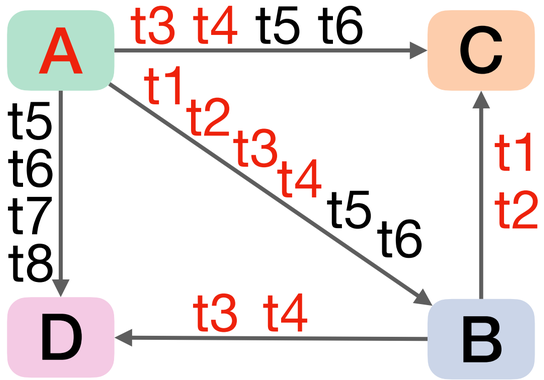


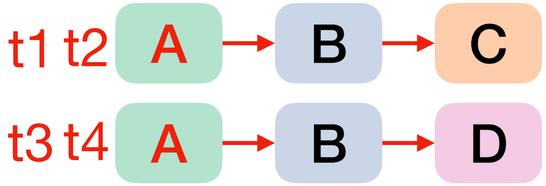


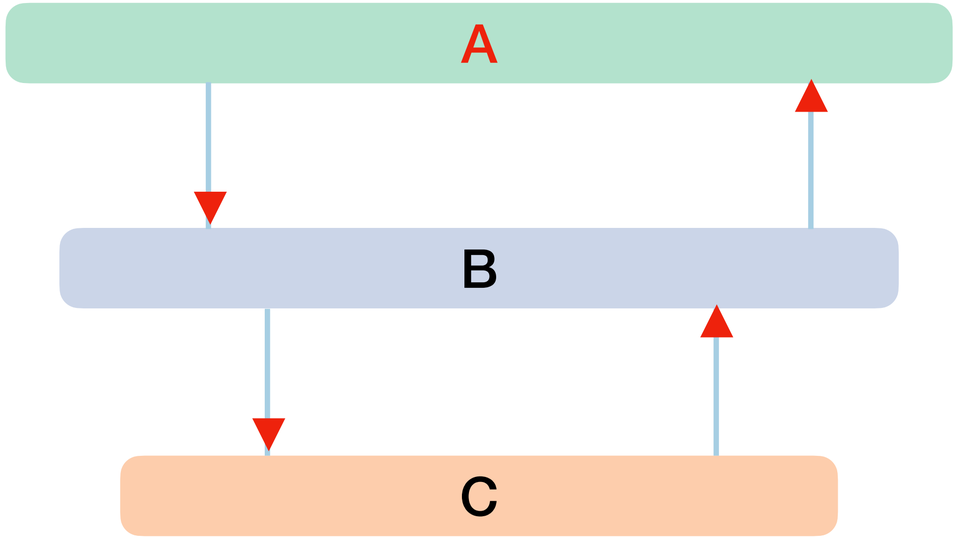
Core Idea

3. Microservice Ranking

Highlight 2: For any service on an abnormal trace, if it has both in-coming and out-coming abnormal invocations, it is probably just propagating the anomaly.



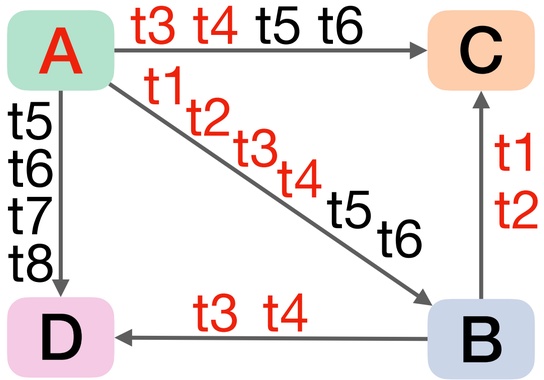


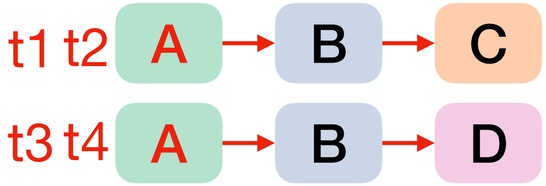


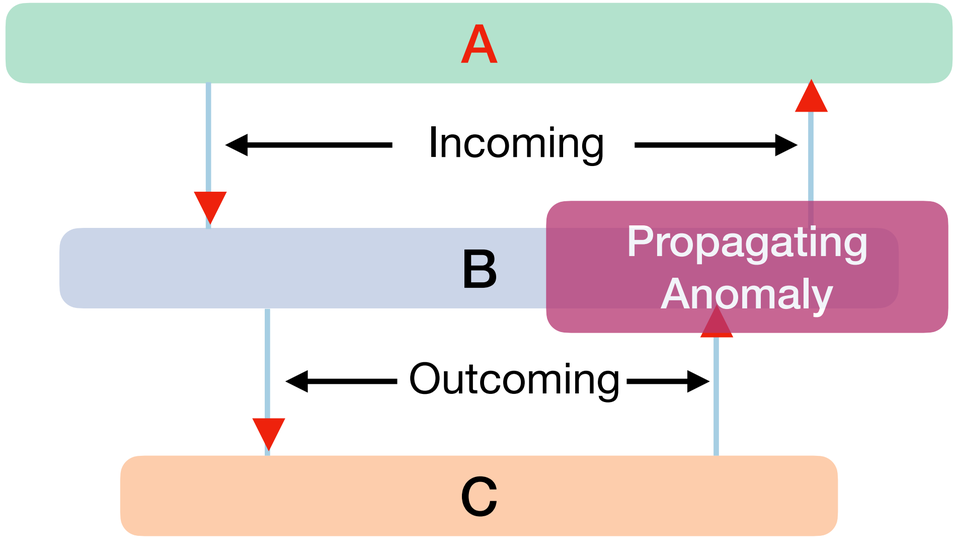
Core Idea

3. Microservice Ranking

Highlight 2: For any service on an abnormal trace, if it has both in-coming and out-coming abnormal invocations, it is probably just propagating the anomaly.



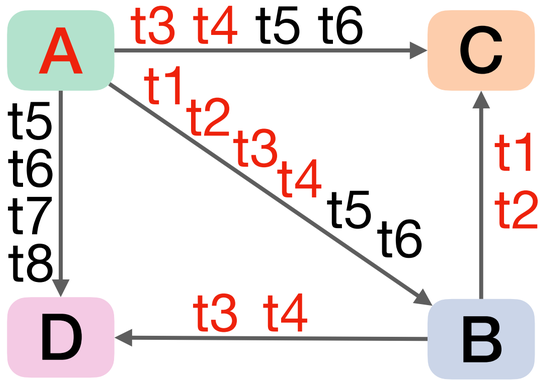


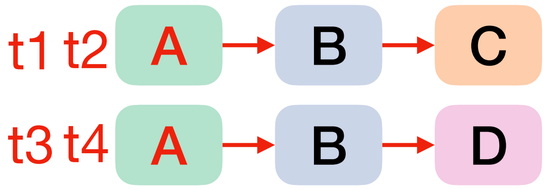


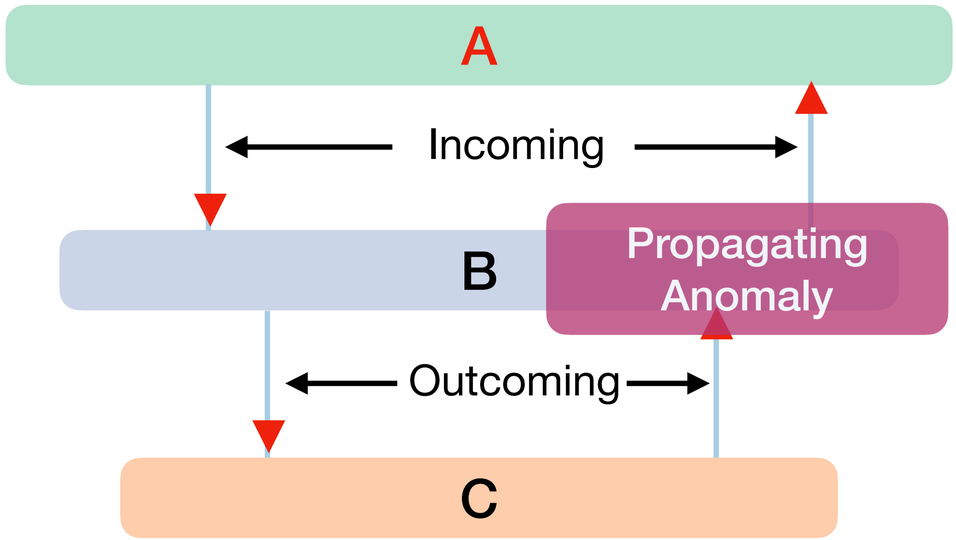
Core Idea

3. Microservice Ranking

Highlight 2: For any service on an abnormal trace, if it has both in-coming and out-coming abnormal invocations, it is probably just propagating the anomaly.







We rank the services inside one suspicious microservice set
by abs(#incoming abnormal invocations - #outcoming abnormal invocations)

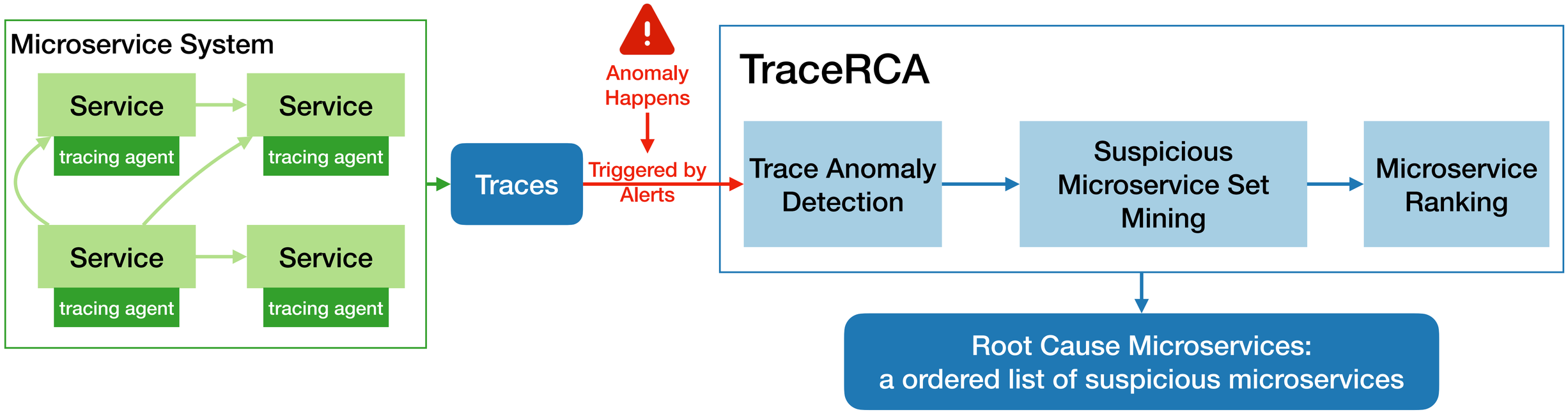
TraceRCA Framework

TraceRCA Framework

TraceRCA: Trace-Based Root Cause Analyzer

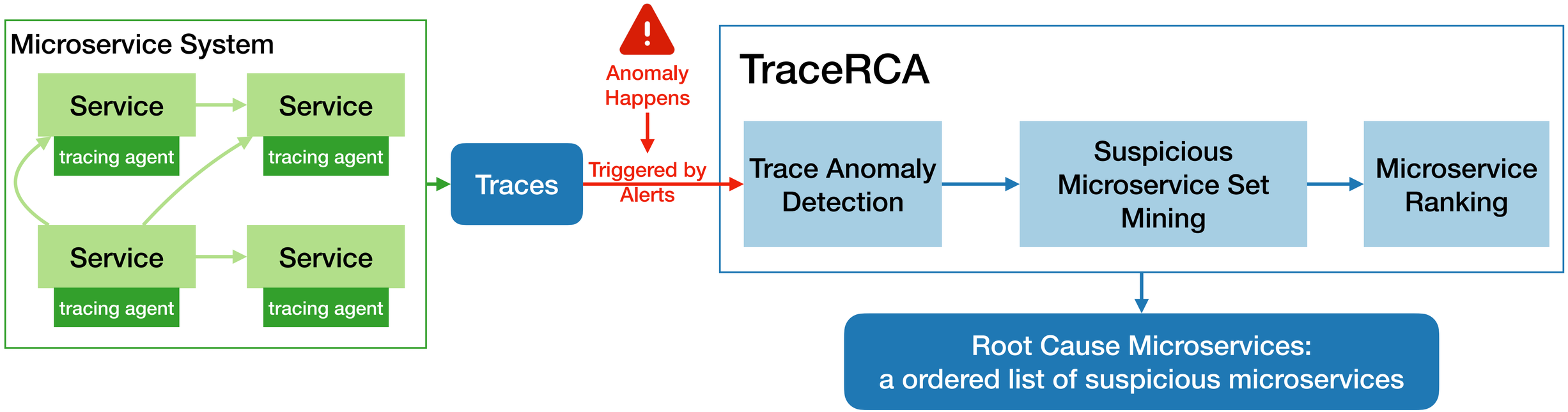
TraceRCA Framework

TraceRCA: Trace-Based Root Cause Analyzer



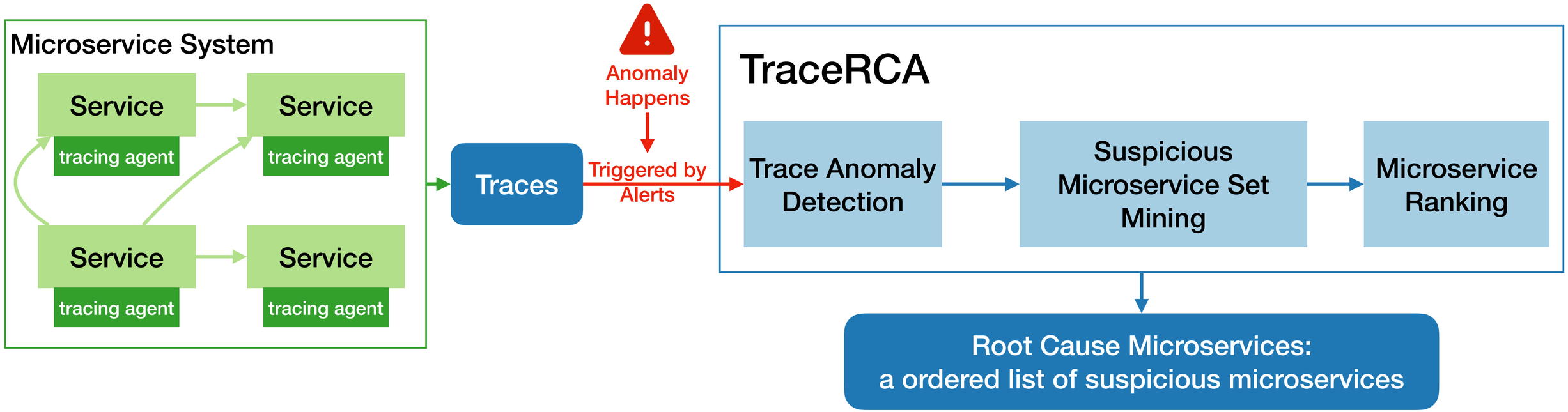
TraceRCA Framework

TraceRCA: Trace-Based Root Cause Analyzer



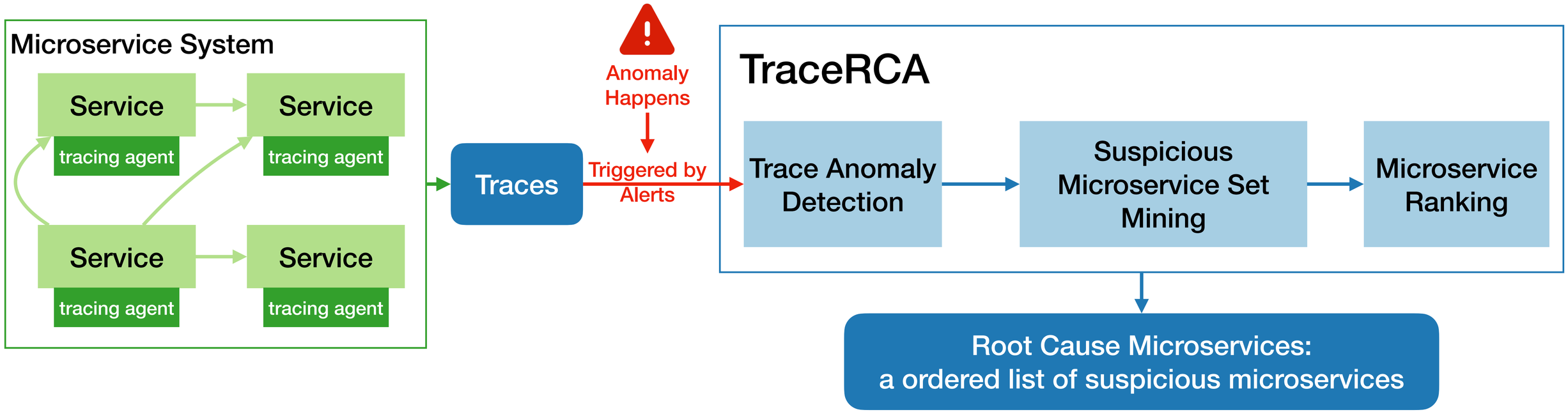
TraceRCA Framework

TraceRCA: Trace-Based Root Cause Analyzer



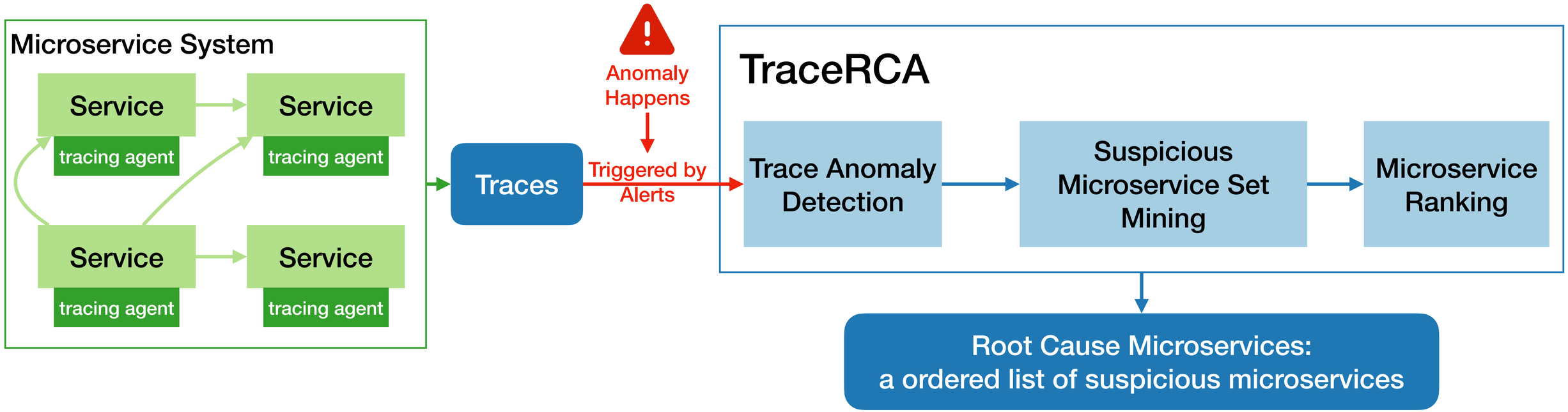
TraceRCA Framework

TraceRCA: Trace-Based Root Cause Analyzer



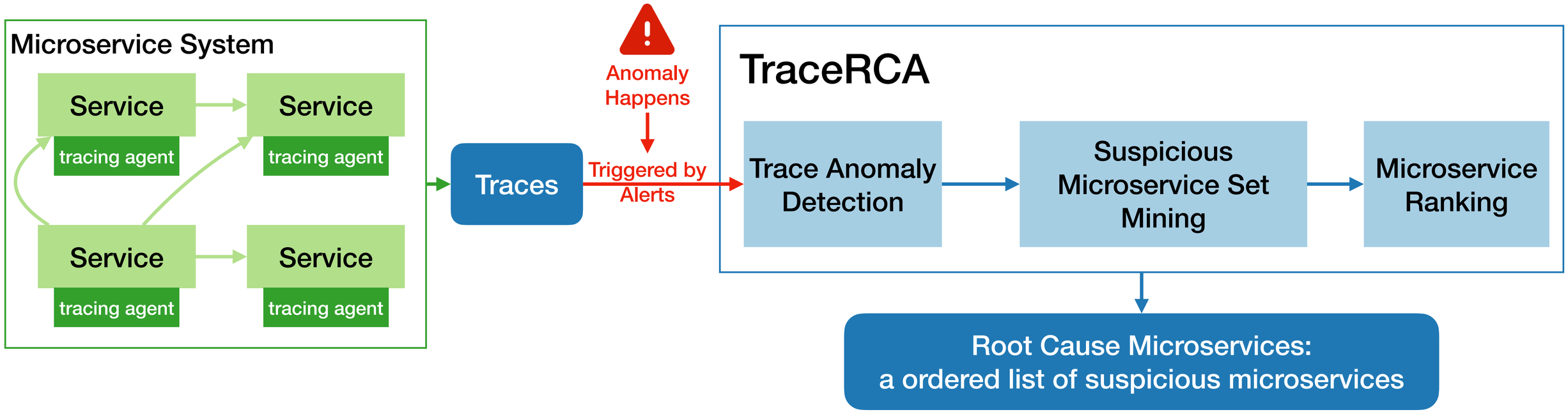
TraceRCA Framework

TraceRCA: Trace-Based Root Cause Analyzer



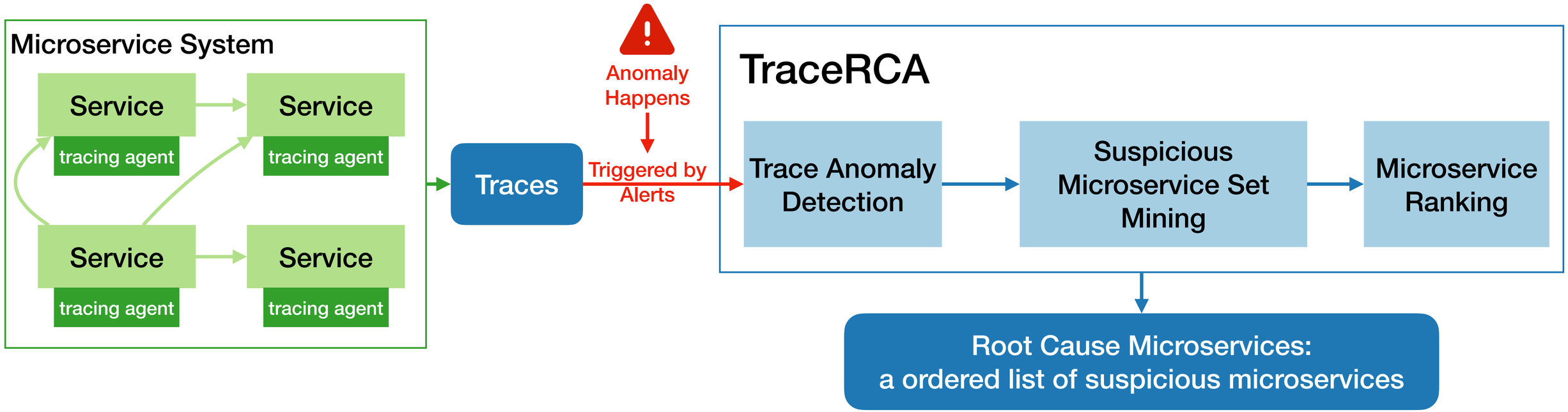
TraceRCA Framework

TraceRCA: Trace-Based Root Cause Analyzer



TraceRCA Framework

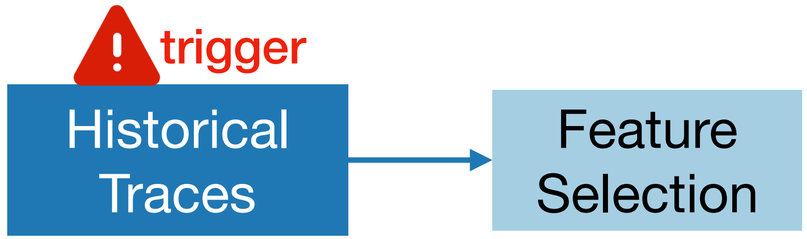
TraceRCA: Trace-Based Root Cause Analyzer



TraceRCA Framework

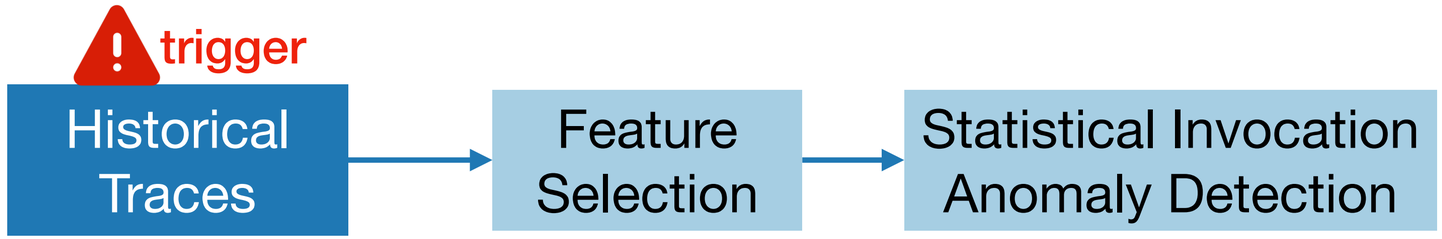
TraceRCA Framework

TraceRCA Framework



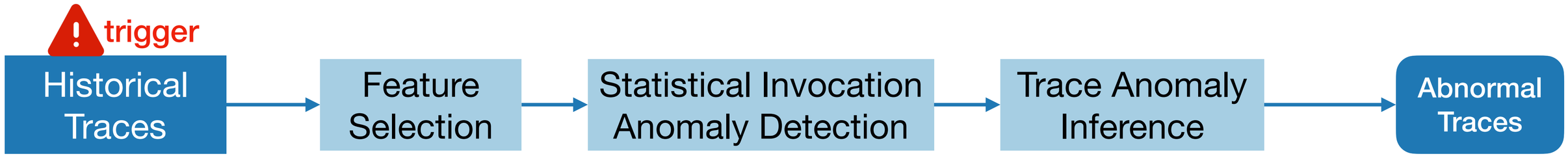
see details in our paper

TraceRCA Framework



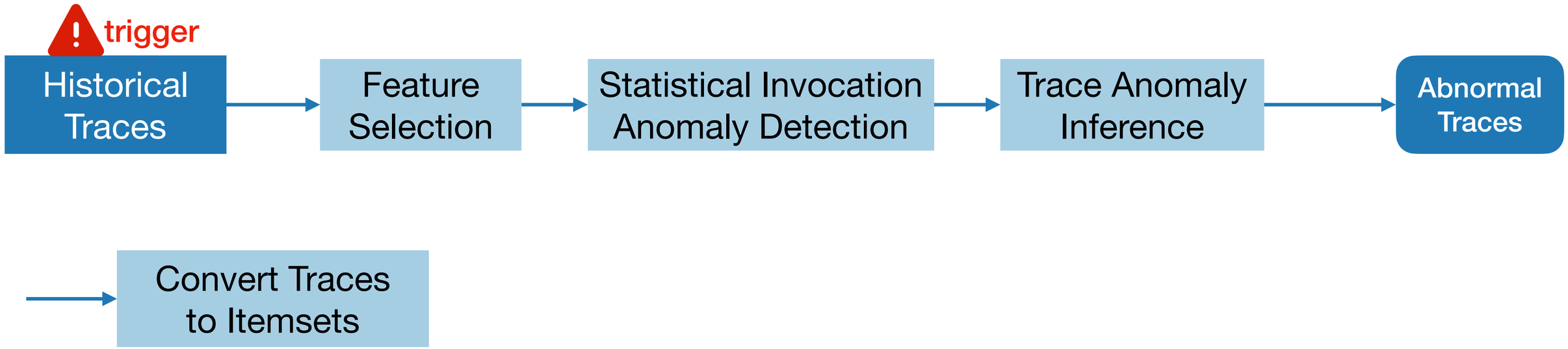
see details in our paper

TraceRCA Framework



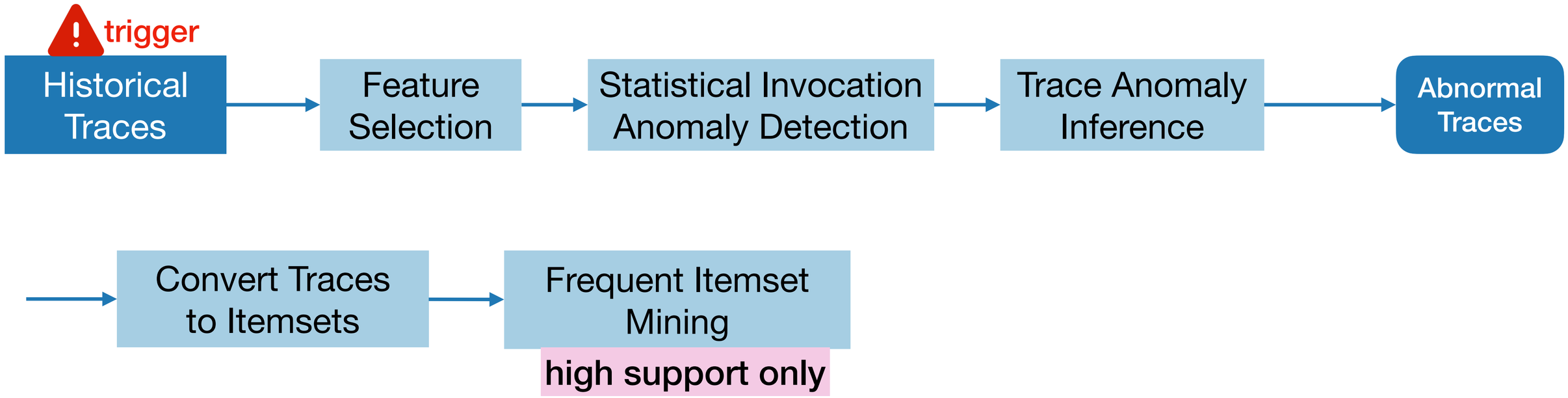
see details in our paper

TraceRCA Framework



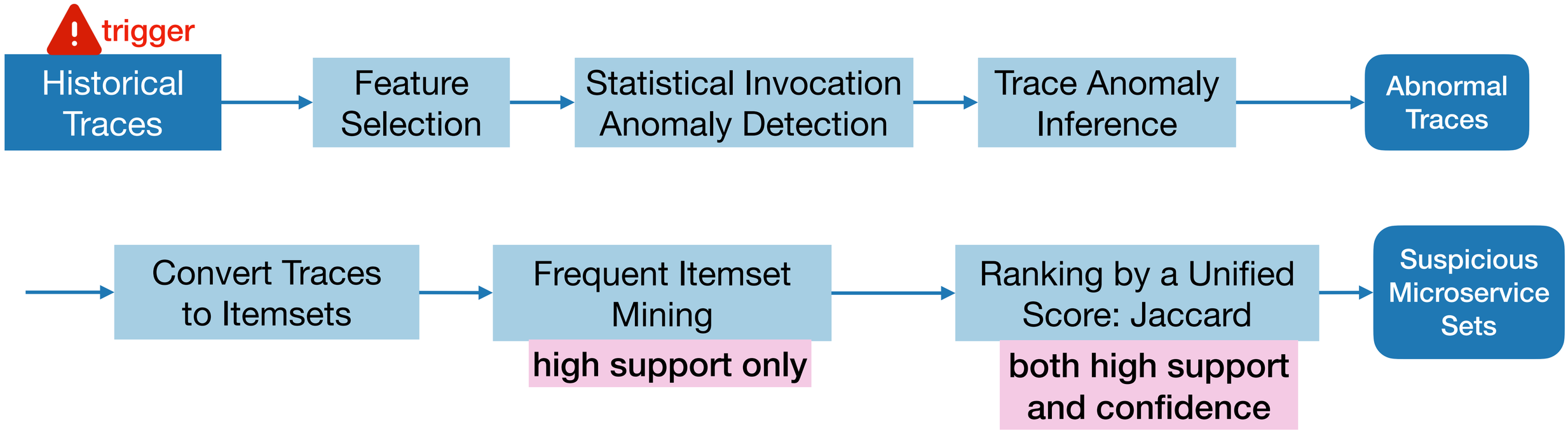
see details in our paper

TraceRCA Framework



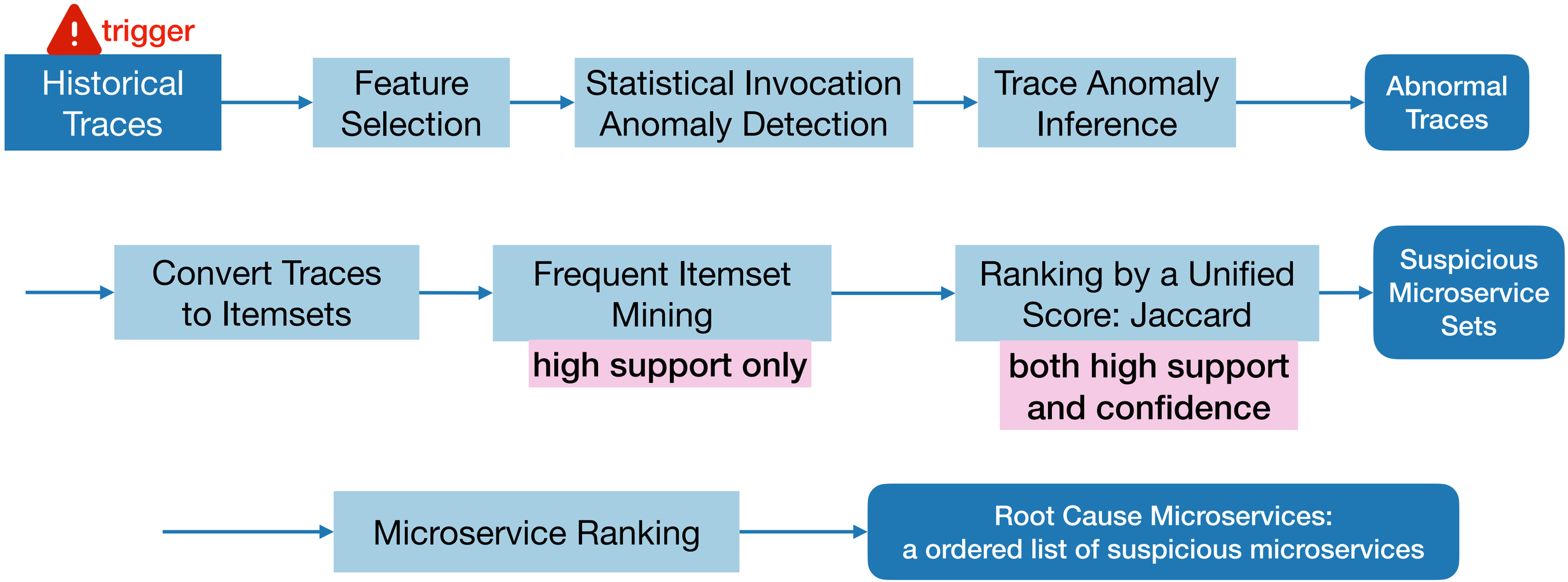
see details in our paper

TraceRCA Framework



see details in our paper

TraceRCA Framework



see details in our paper

Evaluation

Datasets

Dataset	Microservice Benchmark	Fault Type	Root-Cause Component Level	# Faults
A	Train-Ticket [TSE'18]	Application Bug	Microservice	58
		CPU exhausted	Microservice	59
		Network Delay	Microservice	59
		Network Delay	Container	10
		Network Delay	API	14
B	Production System	5 types provided by the developers	-	22

1. There are 11 faults contains more than root causes.
2. Our dataset is now public at https://github.com/netmanaiops/tracerca

Evaluation

Datasets

Dataset	Microservice Benchmark	Fault Type	Root-Cause Component Level	# Faults
A	Train-Ticket [TSE'18]	Application Bug	Microservice	58
		CPU exhausted	Microservice	59
		Network Delay	Microservice	59
		Network Delay	Container	10
		Network Delay	API	14
B	Production System	5 types provided by the developers	-	22

1. There are 11 faults contains more than root causes.
2. Our dataset is now public at https://github.com/netmanaiops/tracerca

Evaluation

Datasets

Dataset	Microservice Benchmark	Fault Type	Root-Cause Component Level	# Faults
A	Train-Ticket [TSE'18]	Application Bug	Microservice	58
		CPU exhausted	Microservice	59
		Network Delay	Microservice	59
		Network Delay	Container	10
		Network Delay	API	14
B	Production System	5 types provided by the developers	-	22

1. There are 11 faults contains more than root causes.
2. Our dataset is now public at https://github.com/netmanaiops/tracerca

Evaluation

Datasets

Dataset	Microservice Benchmark	Fault Type	Root-Cause Component Level	# Faults
A	Train-Ticket [TSE'18]	Application Bug	Microservice	58
		CPU exhausted	Microservice	59
		Network Delay	Microservice	59
		Network Delay	Container	10
		Network Delay	API	14
B	Production System	5 types provided by the developers	-	22

1. There are 11 faults contains more than root causes.
2. Our dataset is now public at https://github.com/netmanaiops/tracerca

Evaluation

Datasets

Dataset	Microservice Benchmark	Fault Type	Root-Cause Component Level	# Faults
A	Train-Ticket [TSE'18]	Application Bug	Microservice	58
		CPU exhausted	Microservice	59
		Network Delay	Microservice	59
		Network Delay	Container	10
		Network Delay	API	14
B	Production System	5 types provided by the developers	-	22

1. There are 11 faults contains more than root causes.
2. Our dataset is now public at https://github.com/netmanaiops/tracerca

Evaluation

Datasets

Dataset	Microservice Benchmark	Fault Type	Root-Cause Component Level	# Faults
A	Train-Ticket [TSE'18]	Application Bug	Microservice	58
		CPU exhausted	Microservice	59
		Network Delay	Microservice	59
		Network Delay	Container	10
		Network Delay	API	14
B	Production System	5 types provided by the developers	-	22

1. There are 11 faults contains more than root causes.
2. Our dataset is now public at https://github.com/netmanaiops/tracerca

Evaluation

Datasets

Dataset	Microservice Benchmark	Fault Type	Root-Cause Component Level	# Faults
A	Train-Ticket [TSE'18]	Application Bug	Microservice	58
		CPU exhausted	Microservice	59
		Network Delay	Microservice	59
		Network Delay	Container	10
		Network Delay	API	14
B	Production System	5 types provided by the developers	-	22

1. There are 11 faults contains more than root causes.
2. Our dataset is now public at https://github.com/netmanaiops/tracerca

Evaluation

Overall Performance

Evaluation

Overall Performance

TABLE III: Comparison of root cause localization on faults of different levels on $\mathcal{A}$

Subject	Algorithm	A@1	⇑A@1	A@2	⇑A@2	A@3	⇑A@3	MAR	⇑MAR	MFR	⇑MFR
Microservice	TraceRCA	0.83	—	0.93	—	0.97	—	1.39	—	1.34	—
	MicroScope	0.56	**46.67%**	0.62	**49.49%**	0.70	**37.33%**	3.64	**61.77%**	3.47	**61.26%**
	MEPFL (RF)	0.94	-12.00%	0.97	-4.82%	0.97	-0.64%	1.42	**1.98%**	1.38	**2.71%**
	Random Walk	0.51	**61.96%**	0.86	**7.25%**	0.94	**3.00%**	1.97	**29.37%**	1.91	**29.51%**
	RCSF	0.52	**60.00%**	0.86	**7.64%**	0.93	**3.69%**	1.68	**16.98%**	1.60	**16.02%**
	TraceAnomaly	0.49	**70.85%**	0.59	**58.14%**	0.63	**53.11%**	4.42	**68.54%**	4.34	**69.13%**
Container	TraceRCA	0.80	—	0.80	—	0.80	—	3.80	0%	3.80	—
	MicroScope	0.20	**300.00%**	0.40	**100.00%**	0.40	**100.00%**	7.20	**47.22%**	7.20	**47.22%**
	MEPFL (RF)	0.80	0.00%	0.80	0.00%	1.00	-20.00%	1.40	-171.43%	1.40	-171.43%
	Random Walk	0.40	**100.00%**	0.60	**33.33%**	0.60	**33.33%**	8.40	**54.76%**	8.40	**54.76%**
	RCSF	0.40	**100.00%**	0.60	**33.33%**	0.60	**33.33%**	3.60	-5.56%	3.60	-5.56%
	TraceAnomaly	0.20	**300.00%**	0.30	**166.67%**	0.30	**166.67%**	7.10	**46.48%**	7.10	**46.48%**
API	TraceRCA	0.83	—	0.83	—	0.83	—	1.75	—	1.75	—
	MicroScope	0.58	**42.86%**	0.67	**64.29%**	0.92	-9.09%	2.00	**12.50%**	2.00	**12.50%**
	MEPFL (RF)	0.83	0.00%	1.00	-16.67%	1.00	-16.67%	1.17	-50.00%	1.17	-50.00%
	Random Walk	0.58	**42.86%**	0.75	**11.11%**	0.64	**29.63%**	2.33	**25.00%**	2.33	**25.00%**
	RCSF	0.42	**100.00%**	0.58	**42.86%**	0.67	**25.00%**	2.58	**32.26%**	2.58	**32.26%**
	TraceAnomaly	0.21	**287.33%**	0.36	**132.40%**	0.50	**66.00%**	5.86	**70.12%**	5.86	**70.12%**

TABLE IV: Comparison of root cause localization on multi-root-cause faults of $\mathcal{A}$

Subject	Algorithm	A@1	⇑A@1	A@2	⇑A@2	A@3	⇑A@3	MAR	⇑MAR	MFR	⇑MFR
multi-root-cause cases on $\mathcal{A}$	TraceRCA	0.45	—	0.82	—	0.95	—	1.77	—	1.09	—
	MicroScope	0.27	**66.67%**	0.27	**200.00%**	0.41	**133.33%**	5.18	**65.79%**	2.73	**60.00%**
	MEPFL (RF)	0.45	0.00%	0.95	-14.29%	0.95	0.00%	1.64	-8.33%	1.09	0.00%
	Random Walk	0.41	**11.11%**	0.64	**28.57%**	0.82	**16.67%**	2.27	**22.00%**	1.36	**20.00%**
	RCSF	0.23	**100.00%**	0.50	**63.64%**	0.73	**31.25%**	2.82	**37.10%**	1.73	**36.84%**
	TraceAnomaly	0.50	**-10.00%**	0.73	**12.75%**	0.82	**16.11%**	2.50	**29.20%**	1.00	**-9.00%**

Evaluation

Overall Performance

TABLE III: Comparison of root cause localization on faults of different levels on $\mathcal{A}$

Subject	Algorithm	A@1	⇑A@1	A@2	⇑A@2	A@3	⇑A@3	MAR	⇑MAR	MFR	⇑MFR
Microservice	TraceRCA	0.83	—	0.93	—	0.97	—	1.39	—	1.34	—
	MicroScope	0.56	**46.67%**	0.62	**49.49%**	0.70	**37.33%**	3.64	**61.77%**	3.47	**61.26%**
	MEPFL (RF)	0.94	-12.00%	0.97	-4.82%	0.97	-0.64%	1.42	**1.98%**	1.38	**2.71%**
	Random Walk	0.51	**61.96%**	0.86	**7.25%**	0.94	**3.00%**	1.97	**29.37%**	1.91	**29.51%**
	RCSF	0.52	**60.00%**	0.86	**7.64%**	0.93	**3.69%**	1.68	**16.98%**	1.60	**16.02%**
	TraceAnomaly	0.49	**70.85%**	0.59	**58.14%**	0.63	**53.11%**	4.42	**68.54%**	4.34	**69.13%**
Container	TraceRCA	0.80	—	0.80	—	0.80	—	3.80	0%	3.80	—
	MicroScope	0.20	**300.00%**	0.40	**100.00%**	0.40	**100.00%**	7.20	**47.22%**	7.20	**47.22%**
	MEPFL (RF)	0.80	0.00%	0.80	0.00%	1.00	-20.00%	1.40	-171.43%	1.40	-171.43%
	Random Walk	0.40	**100.00%**	0.60	**33.33%**	0.60	**33.33%**	8.40	**54.76%**	8.40	**54.76%**
	RCSF	0.40	**100.00%**	0.60	**33.33%**	0.60	**33.33%**	3.60	-5.56%	3.60	-5.56%
	TraceAnomaly	0.20	**300.00%**	0.30	**166.67%**	0.30	**166.67%**	7.10	**46.48%**	7.10	**46.48%**
API	TraceRCA	0.83	—	0.83	—	0.83	—	1.75	—	1.75	—
	MicroScope	0.58	**42.86%**	0.67	**64.29%**	0.92	-9.09%	2.00	**12.50%**	2.00	**12.50%**
	MEPFL (RF)	0.83	0.00%	1.00	-16.67%	1.00	-16.67%	1.17	-50.00%	1.17	-50.00%
	Random Walk	0.58	**42.86%**	0.75	**11.11%**	0.64	**29.63%**	2.33	**25.00%**	2.33	**25.00%**
	RCSF	0.42	**100.00%**	0.58	**42.86%**	0.67	**25.00%**	2.58	**32.26%**	2.58	**32.26%**
	TraceAnomaly	0.21	**287.33%**	0.36	**132.40%**	0.50	**66.00%**	5.86	**70.12%**	5.86	**70.12%**

TABLE IV: Comparison of root cause localization on multi-root-cause faults of $\mathcal{A}$

Subject	Algorithm	A@1	⇑A@1	A@2	⇑A@2	A@3	⇑A@3	MAR	⇑MAR	MFR	⇑MFR
multi-root-cause cases on $\mathcal{A}$	TraceRCA	0.45	—	0.82	—	0.95	—	1.77	—	1.09	—
	MicroScope	0.27	**66.67%**	0.27	**200.00%**	0.41	**133.33%**	5.18	**65.79%**	2.73	**60.00%**
	MEPFL (RF)	0.45	0.00%	0.95	-14.29%	0.95	0.00%	1.64	-8.33%	1.09	0.00%
	Random Walk	0.41	**11.11%**	0.64	**28.57%**	0.82	**16.67%**	2.27	**22.00%**	1.36	**20.00%**
	RCSF	0.23	**100.00%**	0.50	**63.64%**	0.73	**31.25%**	2.82	**37.10%**	1.73	**36.84%**
	TraceAnomaly	0.50	**-10.00%**	0.73	**12.75%**	0.82	**16.11%**	2.50	**29.20%**	1.00	**-9.00%**

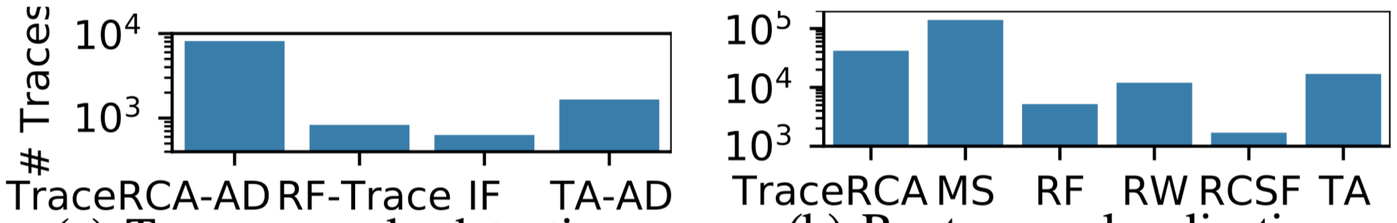
TraceRCA outperforms ***unsupervised*** baselines and perform as well as ***supervised*** approaches under different situations

Evaluation

Scalability

Evaluation

Scalability



(a) Trace anomaly detection (b) Root cause localization

Fig. 13: Comparison of efficiency

TraceRCA handles 10,000 traces per second per core.

Evaluation

Scalability

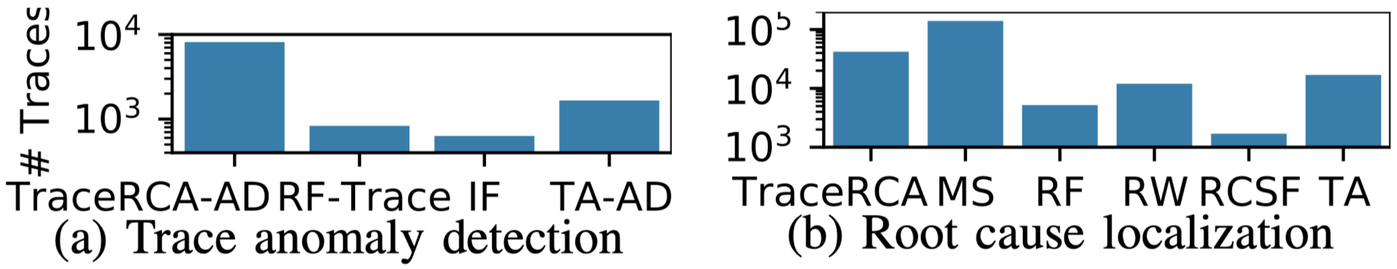


Fig. 13: Comparison of efficiency

TraceRCA handles 10,000 traces per second per core.

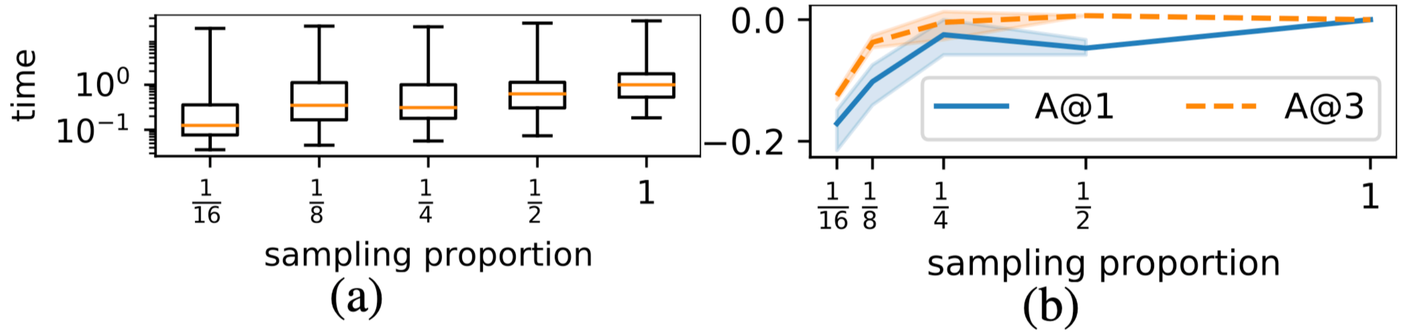


Fig. 14: Efficiency improvement and performance degradation of *TraceRCA* with trace sampling.

Evaluation

Scalability

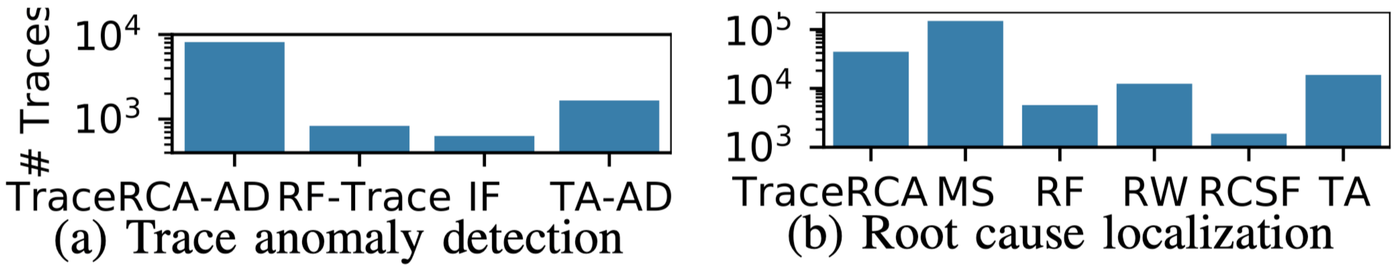


Fig. 13: Comparison of efficiency

TraceRCA handles 10,000 traces per second per core.

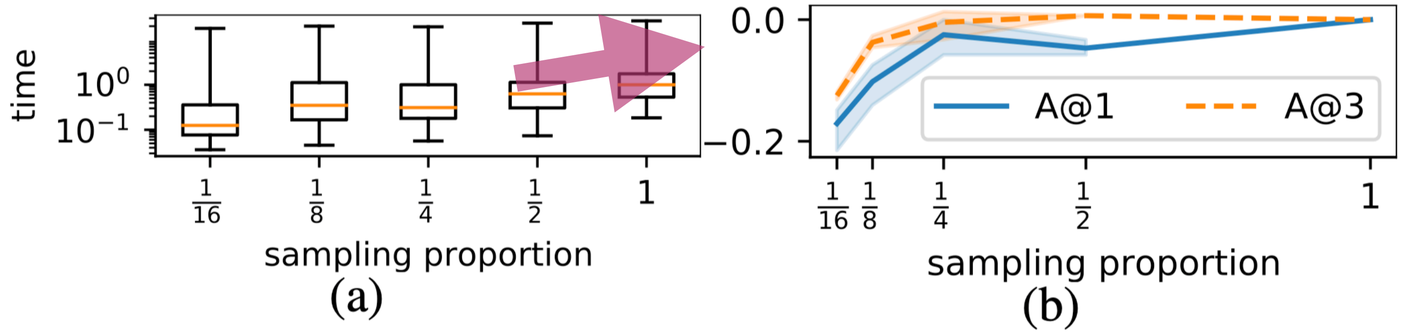


Fig. 14: Efficiency improvement and performance degradation of *TraceRCA* with trace sampling.

The running time of TraceRCA is linear to #traces.

Evaluation

Scalability

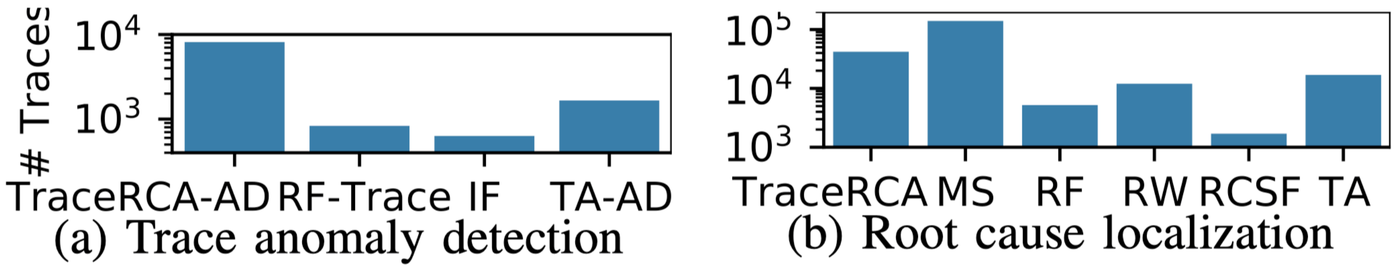


Fig. 13: Comparison of efficiency

TraceRCA handles 10,000 traces per second per core.

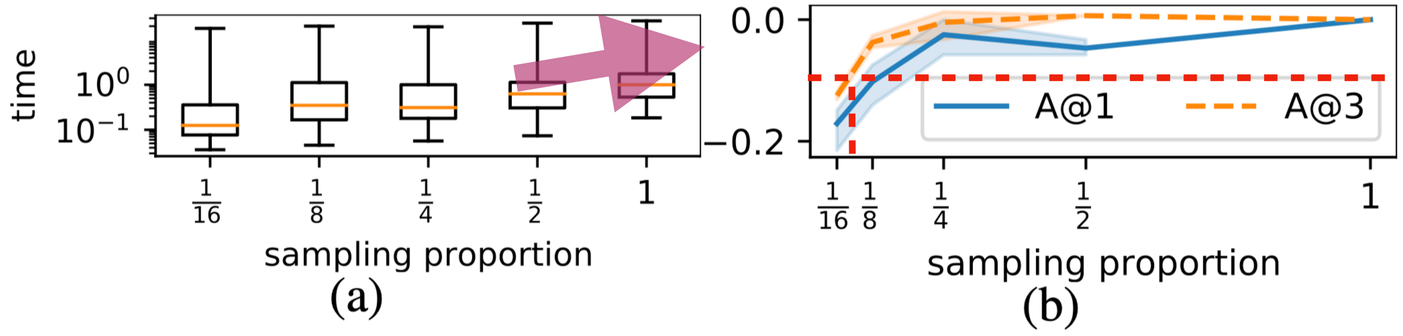


Fig. 14: Efficiency improvement and performance degradation of *TraceRCA* with trace sampling.

The running time of TraceRCA is linear to #traces.

TraceRCA achieves relative good performance with only 1/10 sampled traces.

Summary

Summary

Motivation: Trace-based root-cause service localization; our insight and two key metrics

Summary

Motivation: Trace-based root-cause service localization; our insight and two key metrics

Solution: TraceRCA, containing trace anomaly detection, microservice set mining, suspicious microservice ranking

Summary

Motivation: Trace-based root-cause service localization; our insight and two key metrics

Solution: TraceRCA, containing trace anomaly detection, microservice set mining, suspicious microservice ranking

Evaluation: Experiments based on an open-source benchmark and a production system

Thank You!
Q&A