Localizing Failure Root Causes in a Microservice through Causality Inference

Yuan Meng¹, Shenglin Zhang², Yongqian Sun²
Ruru Zhang², Zhilong Hu², Yiyin Zhang³, Chenyang Jia³, Zhaogang Wang³, Dan Pei¹
Outline

Background  Algorithm  Evaluation  Case Studies
Outline

Background  Algorithm  Evaluation  Case Studies
Failures in Microservice

• **Microservice** has gained an increasing popularity in recent years.

• **Performance quality** of microservice is of vital importance to the Internet company and the users
  
  ➢ On May 18 in 2020, Zoom, experienced a wide range of failures. The COVID-19 official Briefing of British Government was forced to cancel.
  ➢ On March 26 in 2020, Google service broke down for 20 minutes.
  ➢ Netflix reduced stream quality to meet additional demand

• **Efficient root cause localization** of online failures in microservice enables rapid service recovery and loss mitigation.
Microservice architecture

• In the microservice architecture, an application is **decoupled** into multiple microservices.

![Call graph of microservices in the process of placing an order](image)

**Cross-microservices** root cause localization tries to understand how a failure is propagated across microservices and aims to localize the *root cause microservice*, e.g. the Address microservice.

The failure root cause in Address may be the CPU, network, memory, etc.

• In the literature, only the **cross-microservices** root cause localization has been investigated.

• **Failure root causes within a microservice** is still not clear for the operators.
A Microservice

• How does a microservice work?

Fig.1. An example of the microservice

**KPI** (key performance Indicator): a user-perceived indicator that directly reflects the quality of service.

**Metric**: an indicator indicates the status of a microservice’s underlying component.

Potential root causes
Problem Statement

• Failure ticket
  - a microservice ID indicating where the failure occurs
  - a KPI representing which KPI becomes anomalous when this failure occurs
  - a timestamp showing when this failure happens.
  - E.g. {Microservice A, RT of Web, 17:18}

• Problem definition

<table>
<thead>
<tr>
<th>Rank</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Web QPS</td>
</tr>
<tr>
<td>2</td>
<td>JVM YGC Time</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
## Related work

<table>
<thead>
<tr>
<th>Type</th>
<th>Model</th>
<th>Relationship learning</th>
<th>Root cause inference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross-microservice</td>
<td>Microscope[SOC18]</td>
<td>PC</td>
<td>Pearson correlation</td>
</tr>
<tr>
<td></td>
<td>CloudRanger[CCGRID18]</td>
<td>PC</td>
<td>Second order random walk</td>
</tr>
<tr>
<td></td>
<td>MonitorRank[SIGMETRICS13]</td>
<td>Hadoop tools</td>
<td>random walk</td>
</tr>
<tr>
<td></td>
<td>TON18</td>
<td>OpenStack APIs</td>
<td>random walk</td>
</tr>
<tr>
<td>Intra-microservice</td>
<td>MicroCause (our method)</td>
<td>PCTS</td>
<td>TCORW</td>
</tr>
</tbody>
</table>
Challenges

Challenge 1: *iid* based causal graph (e.g. PC) cannot capture propagation delays.

Challenge 2: Correlation based random walk may not accurately localize root cause.

Fig.1: Causal relationship among a KPI and three metrics. A circle denotes a time point of a KPI/metric, and an arrow represents a causal relationship.

Fig.2: Monitoring indicators of failure case {Microservice A, Web RT, 22:45-22:55}
Outline

Background  Algorithm  Evaluation  Case Studies
Model Architecture

Failure ticket X:
{Microservice A, RT of Web, 17:18}

MicroCause
• Failure Causal Graph Learning

  PCTS

**Step 1: Improved PC algorithm[1]**

**Step 2: Generate failure causal graph**

---

MicroCause

Failure Causal Graph Learning

Metrics

+ KPI

Failure
MicroCause

• Anomaly detection
  - SPOT[KDD17]
    - detects the sudden change in time series via the extreme value theory
  - Anomaly degree

\[
\eta_{\text{max}}^i = \max_{k \in O} \frac{|M_k^i - \phi M_k^i|}{\phi M_k^i}
\]

O is the index set of the anomaly point, \( \phi M_k^i \) is the threshold
MicroCause

Failure Causal Graph Learning

KPI +

Metrics

Anomaly detection

<table>
<thead>
<tr>
<th>Metrics</th>
<th>If Anomalous</th>
<th>Anomaly Time</th>
<th>Anomaly Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>QPS of Web</td>
<td>Yes</td>
<td>17:17</td>
<td>10.62</td>
</tr>
<tr>
<td>YGC Time of JVM</td>
<td>Yes</td>
<td>17:17</td>
<td>18.35</td>
</tr>
<tr>
<td>RT of Middleware1</td>
<td>Yes</td>
<td>17:18</td>
<td>2.40</td>
</tr>
<tr>
<td>Memory Utilization of System</td>
<td>No</td>
<td>Not Exist</td>
<td>Not Exist</td>
</tr>
</tbody>
</table>
• Temporal Cause Oriented Random Walk (TCORW)
  ➢ Step one: Cause oriented random walk (causal relationship)
  ➢ Step two: Potential root cause score (+ anomaly degree)
  ➢ Step three: Rank the root causes (+ priority, anomaly time)
MicroCause

- **Temporal Cause Oriented Random Walk (TCORW)**
  - **Step one:** Cause oriented random walk \((\text{causal relationship})\)
    - **Partial correlation**
    - Forward step (walk from effect indicator to cause indicator)
      \[
      Q_{ij} = R_{pc}(v_{ak}, v_j | Pa(v_{ak}) \setminus v_j, Pa(v_j))
      \]
    - Backward step (walk from cause indicator to effect indicator)
      \[
      Q_{ji} = \rho R_{pc}(v_{ak}, v_i | Pa(v_{ak}) \setminus v_i, Pa(v_i))
      \]
    - Self step (stay in the present node):
      \[
      Q_{ii} = \max[0, R_{pc}(v_{ak}, v_i | Pa(v_{ak}) \setminus v_i, Pa(v_i)) - P_{pc}^{max}] \\
      P_{pc}^{max} = \max_{k: e_k \in E} R_{pc}(v_{ak}, v_k | Pa(v_{ak}) \setminus v_k, Pa(v_k))
      \]
MicroCause

• Temporal Cause Oriented Random Walk (TCORW)
  ➢ Step two: Potential root cause score (+ anomaly degree)

\[ \gamma_i = \lambda \bar{c}_i + (1 - \lambda) \bar{\eta}_{max}^i \]

\(\bar{c}_i\) is the normalized visit time \(c_i\). \(\bar{\eta}_{max}^i\) is the normalized anomaly degree \(\eta_{max}^i\). \(\lambda\) controls the contribution of metric’s causal relationship with the anomalous KPI and the anomaly degree of the metric.
### MicroCause

- **Temporal Cause Oriented Random Walk (TCORW)**
  - Step three: Rank the root causes (+ priority, anomaly time)

<table>
<thead>
<tr>
<th>Metrics Type</th>
<th>Metrics</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upstream</td>
<td>QPS of Web; Provider QPS of middleware1; Receive QPS of middleware2; Receive QPS of middleware3</td>
<td>Level 1</td>
</tr>
<tr>
<td>Self</td>
<td><strong>Java virtual machine (JVM) related</strong>: YGC count of JVM; YGC time of JVM; FGC count of JVM; FGC time of JVM; Max heap memory of JVM; Used heap memory of JVM; Usage heap memory of JVM; Max nonheap memory of JVM; Used nonheap memory of JVM; Usage metaspace pools of JVM memory; Usage code cache pools of JVM memory; Max mapped bufferpool of JVM; Used mapped bufferpool of JVM; Max direct bufferpool of JVM; Used direct bufferpool of JVM; Thread count of JVM; Deamon thread count of JVM; Deadlock thread count of JVM; Runnable thread count of JVM; File descriptor utilization of JVM;</td>
<td>Level 2</td>
</tr>
<tr>
<td>Downstream</td>
<td><strong>System related</strong>: CPU utilization of system; CPU steal of system; Load1 utilization of system; Load5 utilization of system; Load15 utilization of system; Load1 of system; Load5 of system; Load15 of system; Memory utilization of system; Swap utilization of system; Net in of system; Net out of system; Net retran utilization of system; Net established of system; Disk utilization of system; Disk read of system; Disk write of system; Disk inode of system;</td>
<td>Level 2</td>
</tr>
</tbody>
</table>

- Queries per second (QPS): Consumer QPS of middleware1; Read QPS of middleware4; Write QPS of middleware1; Read QPS of middleware5; Write QPS of middleware5; Send QPS of middleware2; Send QPS of middleware3;
- Response time (RT): Consumer RT of middleware1; Read RT of middleware4; Write RT of middleware4; Read RT of middleware5; Write RT of middleware5; Send RT of middleware2; Send RT of middleware3;
- Success rate: Consumer success rate of middleware1; Read success rate of middleware4; Write success rate of middleware4; Read success rate of middleware5; Write success rate of middleware5; Send success rate of middleware2; Send success rate of middleware3;
• Temporal Cause Oriented Random Walk (TCORW)

  Step three: Rank the root causes (+ priority, anomaly time)

```
Algorithm 1: Rank the root cause

Input: 1 Levels of metrics, 2 \gamma_i of metric i, 3 anomaly time t_i of metric i

Output: RankResultSet

ResultSet ← []

for j = 1, 2, 3 do

  \( R_j \leftarrow \text{rank metrics in Level j by } \gamma_i \text{ in descending order.} \)

  ResultSet ← append the top 2 result in \( R_j \)

end

RankResultSet ← rank ResultSet by \( t_i \) in ascending order
```
MicroCause

Failure ticket X: {Microservice A, RT of Web, 17:18}

Failure Causal Graph Learning

Temporal Cause Oriented Random Walk

Top N Root Cause List

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Anomaly Time</th>
<th>Anomaly Degree</th>
<th>Level</th>
<th>Visit Time</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>QPS of Web</td>
<td>17:17</td>
<td>10.62</td>
<td>Level 1</td>
<td>270</td>
<td>1</td>
</tr>
<tr>
<td>YGC Time of JVM</td>
<td>17:17</td>
<td>18.35</td>
<td>Level 2</td>
<td>75</td>
<td>2</td>
</tr>
<tr>
<td>RT of Middleware1</td>
<td>17:18</td>
<td>2.40</td>
<td>Level 3</td>
<td>180</td>
<td>3</td>
</tr>
<tr>
<td>Memory Utilization</td>
<td>No</td>
<td>Not Exist</td>
<td>Not Exist</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>of System</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outline

Background  Algorithm  Evaluation  Case Studies
Dataset & Evaluation Metrics

• Dataset
  - 86 online failure tickets in an online shopping platform
  - monitoring more than 400 microservice status.
  - Sep. 2019 to Jan. 2020
  - 4 KPIs:
    - RT of Web
    - provider RT of middleware1
    - receive RT of middleware2
    - receive RT of middleware3.

• Evaluation Metrics

\[
AC@k = \frac{1}{|A|} \sum_{a \in A} \sum_{i \leq k} \frac{R^a[i] \in V^a_{rc}}{\min(k, |V^a_{rc}|)}
\]

\[
Avg@k = \frac{1}{k} \sum_{1 \leq j \leq k} AC@j
\]
## MicroCause VS baseline methods

<table>
<thead>
<tr>
<th>Method</th>
<th>AC@1</th>
<th>AC@2</th>
<th>AC@5</th>
<th>Avg@5</th>
</tr>
</thead>
<tbody>
<tr>
<td>MicroCause</td>
<td>46.7%</td>
<td>62.7%</td>
<td>98.7%</td>
<td>69.7%</td>
</tr>
<tr>
<td>TON18, MonitorRank[SIGMETRICS13]</td>
<td>34.7%(−12.0%)</td>
<td>48.0%(−14.7%)</td>
<td>65.3%(−33.4%)</td>
<td>48.2%(−21.5%)</td>
</tr>
<tr>
<td>CloudRanger[CCGGRID18]</td>
<td>19.0%</td>
<td>32.9%</td>
<td>69.6%</td>
<td>46.8%</td>
</tr>
<tr>
<td>Microscope[SOC18]</td>
<td>12.2%</td>
<td>21.9%</td>
<td>29.3%</td>
<td>23.9%</td>
</tr>
<tr>
<td>Anomaly Time Order</td>
<td>11.4%</td>
<td>21.5%</td>
<td>43.0%</td>
<td>28.4%</td>
</tr>
</tbody>
</table>
Analysis about MicroCause

- **Evaluation of PCTS**

<table>
<thead>
<tr>
<th>Method</th>
<th>AC@1</th>
<th>AC@2</th>
<th>AC@5</th>
<th>Avg@5</th>
</tr>
</thead>
<tbody>
<tr>
<td>MicroCause</td>
<td>46.7%</td>
<td>62.7%</td>
<td>98.7%</td>
<td>69.7%</td>
</tr>
<tr>
<td>MicroCause w/PC</td>
<td>44.9%(-1.8%)</td>
<td>59.0%(-3.7%)</td>
<td>93.6%(-5.1%)</td>
<td>67.4%(-2.3%)</td>
</tr>
</tbody>
</table>

- **Evaluation of TCORW**

<table>
<thead>
<tr>
<th>Method</th>
<th>AC@1</th>
<th>AC@2</th>
<th>AC@5</th>
<th>Avg@5</th>
</tr>
</thead>
<tbody>
<tr>
<td>MicroCause</td>
<td>46.7%</td>
<td>62.7%</td>
<td>98.7%</td>
<td>69.7%</td>
</tr>
<tr>
<td>MicroCause w/RW-1</td>
<td>34.7%(-12.0%)</td>
<td>48.0%(-14.7%)</td>
<td>65.3%(-33.4%)</td>
<td>48.2%(-21.5%)</td>
</tr>
<tr>
<td>MicroCause w/RW-2</td>
<td>29.3%</td>
<td>46.7%</td>
<td>62.7%</td>
<td>46.3%</td>
</tr>
</tbody>
</table>
Analysis about MicroCause

- Parameters in MicroCause
Outline

Background  Algorithm  Evaluation  Case Studies
Causal graph of time series

- Isolated subgraphs via PC[INFOCOM14]

**Fig1:** Failure causal graph via PC algorithm of failure ticket X
{Microservice A, RT of Web, 17:18}

**Fig2:** Failure causal graph via PCTS algorithm of failure ticket X
{Microservice A, RT of Web, 17:18}
Conclusion

• To the best of our knowledge, this paper is the first attempt to investigate the failure root cause in a microservice.

• We design a framework, MicroCause, to localize the failure root cause in a microservice, which achieves high performance in the experiments based on 86 the online failure tickets.

• In MicroCause, we design PCTS, which can learn the causal graph of monitoring indicators. We believe it can be used in other time series related root cause localization problems.
Thank you!

Q & A

meng-y16@mails.tsinghua.edu.cn
IWQOS 2020