FluxInfer: Automatic Diagnosis of Performance Anomaly for Online Database System

Ping Liu¹, Shenglin Zhang², Yongqian Sun², Yuan Meng¹, Jiahai Yang¹,³, Dan Pei¹
Background  Design  Evaluation  Case Study
Background  Design  Evaluation  Case Study
The services need databases to support their mission-critical and real-time applications.
Database anomaly

Cannot open webpage!

Response Time Anomaly
Database anomaly

How to diagnose the anomalies?
Anomaly diagnosis

• To rapidly diagnose anomalies and trigger mitigation, database operators monitor hundreds of KPIs (Key Performance Indicator) of the database.
Anomaly diagnosis

- To rapidly diagnose anomalies and trigger mitigation, database operators monitor hundreds of KPIs (Key Performance Indicator) of the database.

**Workload related KPIs**
- Query requests: INSERT, UPDATE, DELETE, ...
Anomaly diagnosis

- To rapidly diagnose anomalies and trigger mitigation, database operators monitor hundreds of KPIs (Key Performance Indicator) of the database.

**Workload related KPIs**
- Query requests: INSERT, UPDATE, DELETE, ...

**Storage-Engine-IO related KPIs**
- **Physical IO**: The actual number of physical bytes involved in IO related operations.
- **Logic IO**: The actual number of virtual pages involved in IO related operations.
Anomaly diagnosis

- To rapidly diagnose anomalies and trigger mitigation, database operators monitor hundreds of KPIs (Key Performance Indicator) of the database.

- **Workload related KPIs**
  - Query requests: INSERT, UPDATE, DELETE, ...

- **Resource related KPIs**
  - CPU, Memory, Network, ...

- **Storage-Engine-IO related KPIs**
  - **Physical IO**: The actual number of physical bytes involved in IO related operations.
  - **Logic IO**: The actual number of virtual pages involved in IO related operations.
Anomaly diagnosis

• To rapidly diagnose anomalies and trigger mitigation, database operators monitor hundreds of KPIs (Key Performance Indicator) of the database.

**Workload related KPIs**
- Query requests: INSERT, UPDATE, DELETE, ...

**Resource related KPIs**
- CPU, Memory, Network, ...

**Storage-Engine-IO related KPIs**
- **Physical IO**: The actual number of physical bytes involved in IO related operations.
- **Logic IO**: The actual number of virtual pages involved in IO related operations.

**Business related KPIs**
- Response time, ...
Anomaly diagnosis

- When a database anomaly arises (response times are too slowly), some KPIs manifest anomalous patterns.
Anomaly diagnosis

DB

Anomalous patterns
Anomaly diagnosis

Root cause related KPI
• These KPIs indicate the root cause of the database anomaly and thus is closely related to the root cause

Symptom KPI
• These KPIs are not directly related to the root cause
Anomaly diagnosis

**Root cause related KPI**
- These KPIs indicate the root cause of the database anomaly and thus is closely related to the root cause

**Symptom KPI**
- These KPIs are not directly related to the root cause

**Example**
- A sudden increase in workload leads to a database anomaly.
- The QPS KPI, which represents the workload of the database, is a **root cause related KPI**.
- On the other hand, the increase in workload also leads to a sudden increase in CPU utilization, which is represented by the CPU\_USAGE KPI. Consequently, CPU\_USAGE KPI is a **symptom KPI**.
Anomaly diagnosis

Which are the root cause related KPIs
Anomaly diagnosis

Operators manually infer the dependency relationships of these KPIs, which is highly dependent on the domain knowledge and experience.
Anomaly diagnosis

Then localize the root cause related KPIs based on the dependency relationships
Anomaly diagnosis

However, it is **time-consuming** to manually diagnose a database anomaly even for a **highly experienced** operator, causing the services to suffer from performance degradation for a long period.

Then localize the root cause related KPIs based on the dependency relationships.
Core idea
FluxInfer can automatically construct the dependency relationships of anomalous KPIs accurately and localize root cause related KPIs.
Design

Services

Databases

Response Time
Anomaly

Performance Alert

FluxInfer
Design

Services → Databases

Performance Alert

Response Time

Anomaly

FluxInfer

Step-1: Anomaly Detection

KPIs

Time Series Database
Design

Step 1: Anomaly Detection

Step 2: Dependency Graph Construction

FluxInfer

Services

Databases

Response Time

Anomaly

Performance
Alert

DB

Time Series
Database

KPIs

Anomalous KPIs
Dependency Graph

Weighted undirected dependency graph (WUDG)
The strength of the dependency between two KPIs can be measured by: \( \frac{1}{p_{\text{value}_4^3}} \)
Design

Services

Databases

Performance Alert

Response Time Anomaly

FluxInfer

Step-1: Anomaly Detection

KPIs

Time Series Database

Anomalous KPIs

Step-2: Dependency Graph Construction

WUDG

Diagnosis Result

Step-3: Root Cause Localization

Anomaly

FluxInfer
Root Cause Localization

• Due to that the root cause of a database system can quickly spread and lead to more and more anomalous KPIs, the root cause has the largest influence on the dependency graph among KPIs.

• Therefore, we suppose that the root cause related KPI is the KPI who has the largest influence on the dependency graph.

• The weighted PageRank algorithm are used to measure the influences of nodes in a weighted undirected graph, the possible root cause related KPIs are the KPIs ranked at the top.
Design

FluxInfer

**Step-1:** Anomaly Detection

Mitigation Method
- SQL flow control
- SQL optimization
- Auto-scale
- ...

**Step-2:** Dependency Graph Construction

WUDG

**Step-3:** Root Cause Localization

KPIs

Anomalous KPIs

Time Series Database

Operator

Diagnosis Result

Mitigation

Performance Alert

Response Time Anomaly

Databases

Services
Dataset

- We constructed a testbed to generate accurately labeled anomalies of database performance for evaluation.
- We injected five different types of anomalies: CPU saturation, network congestion, IO saturation, memory saturation, anomalous workload.

<table>
<thead>
<tr>
<th>Type of anomaly</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Saturation</td>
<td>30</td>
<td>Invoke stress-ng, which starts N workers that perform various matrix operations on floating point values.</td>
</tr>
<tr>
<td>Network Congestion</td>
<td>30</td>
<td>Simulate network congestion by adding an artificial 500-millisecond delay to every traffic over the network via Linux’s tc (Traffic Control) command.</td>
</tr>
<tr>
<td>IO Saturation</td>
<td>30</td>
<td>Invoke stress-ng, which starts N workers that perform a mix of sequential, random and memory mapped read/write operations as well as forced sync’ing and cache dropping.</td>
</tr>
<tr>
<td>Memory Saturation</td>
<td>30</td>
<td>Invoke stress-ng, which starts N workers that grow their heaps by reallocating memory.</td>
</tr>
<tr>
<td>Anomalous workload</td>
<td>30</td>
<td>Greatly increase the rate of transactions and the number of clients simulated by OLTPBenchmark (150 additional terminals with transaction rate of 50,000).</td>
</tr>
</tbody>
</table>
Evaluation metric

- **AC@k** represents the accuracy that top K results include the root causes related KPIs for all anomaly cases.
- **Avg@k** represents the overall performance.

\[
Avg@k = \frac{1}{k} \sum_{1 \leq j \leq k} AC@j.
\]
### TABLE III: The evaluation results of different algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Relationships Learning</th>
<th>Root Cause Inference</th>
<th>AC@1</th>
<th>AC@2</th>
<th>AC@3</th>
<th>AC@5</th>
<th>Avg@5</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAL [20]</td>
<td>N/A</td>
<td>Anomaly Time Order</td>
<td>0.09</td>
<td>0.12</td>
<td>0.14</td>
<td>0.20</td>
<td>0.14</td>
</tr>
<tr>
<td>CauseInfer [3]</td>
<td>PC Algorithm</td>
<td>Deep First Search</td>
<td>0.12</td>
<td>0.20</td>
<td>0.22</td>
<td>0.28</td>
<td>0.21</td>
</tr>
<tr>
<td>CloudRanger [4], MS-Rank [7]</td>
<td>PC Algorithm</td>
<td>Second-order Random Walk</td>
<td>0.08</td>
<td>0.19</td>
<td>0.27</td>
<td>0.36</td>
<td>0.24</td>
</tr>
<tr>
<td>Microscope [6]</td>
<td>PC Algorithm</td>
<td>Traversing+Pearson Correlation</td>
<td>0.06</td>
<td>0.11</td>
<td>0.16</td>
<td>0.24</td>
<td>0.15</td>
</tr>
<tr>
<td>MicroRCA [21]</td>
<td>PC Algorithm</td>
<td>Personalized PageRank</td>
<td>0.08</td>
<td>0.17</td>
<td>0.30</td>
<td>0.38</td>
<td>0.25</td>
</tr>
<tr>
<td>MonitorRank [18], TON18 [19]</td>
<td>PC Algorithm</td>
<td>Random Walk</td>
<td>0.08</td>
<td>0.16</td>
<td>0.28</td>
<td>0.39</td>
<td>0.24</td>
</tr>
<tr>
<td>MicroCause [5]</td>
<td>PCTS</td>
<td>TCORW</td>
<td>0.23</td>
<td>0.38</td>
<td>0.47</td>
<td>0.60</td>
<td>0.44</td>
</tr>
<tr>
<td><strong>FluxInfer</strong></td>
<td>WUDG</td>
<td>Weighted PageRank</td>
<td><strong>0.43</strong></td>
<td><strong>0.69</strong></td>
<td><strong>0.90</strong></td>
<td><strong>0.95</strong></td>
<td><strong>0.77</strong></td>
</tr>
<tr>
<td>FluxInfer-with-CUSUM</td>
<td>WUDG</td>
<td>Weighted PageRank</td>
<td>0.23</td>
<td>0.40</td>
<td>0.62</td>
<td>0.73</td>
<td>0.53</td>
</tr>
<tr>
<td>FluxInfer-without-AD</td>
<td>WUDG</td>
<td>Weighted PageRank</td>
<td>0.13</td>
<td>0.20</td>
<td>0.32</td>
<td>0.50</td>
<td>0.38</td>
</tr>
</tbody>
</table>
### Evaluation result

**TABLE III: The evaluation results of different algorithms**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Relationships Learning</th>
<th>Root Cause Inference</th>
<th>AC@1</th>
<th>AC@2</th>
<th>AC@3</th>
<th>AC@5</th>
<th>Avg@5</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAL [20]</td>
<td>N/A</td>
<td>Anomaly Time Order</td>
<td>0.09</td>
<td>0.12</td>
<td>0.14</td>
<td>0.20</td>
<td>0.14</td>
</tr>
<tr>
<td>CauseInfer [3]</td>
<td>PC Algorithm</td>
<td>Deep First Search</td>
<td>0.12</td>
<td>0.20</td>
<td>0.22</td>
<td>0.28</td>
<td>0.21</td>
</tr>
<tr>
<td>CloudRanger [4], MS-Rank [7]</td>
<td>PC Algorithm</td>
<td>Second-order Random Walk</td>
<td>0.08</td>
<td>0.19</td>
<td>0.27</td>
<td>0.36</td>
<td>0.24</td>
</tr>
<tr>
<td>Microscope [6]</td>
<td>PC Algorithm</td>
<td>Traversing+Pearson Correlation</td>
<td>0.06</td>
<td>0.11</td>
<td>0.16</td>
<td>0.24</td>
<td>0.15</td>
</tr>
<tr>
<td>MicroRCA [21]</td>
<td>PC Algorithm</td>
<td>Personalized PageRank</td>
<td>0.08</td>
<td>0.17</td>
<td>0.30</td>
<td>0.38</td>
<td>0.25</td>
</tr>
<tr>
<td>MonitorRank [18], TON18 [19]</td>
<td>PC Algorithm</td>
<td>Random Walk</td>
<td>0.08</td>
<td>0.16</td>
<td>0.28</td>
<td>0.39</td>
<td>0.24</td>
</tr>
<tr>
<td>MicroCause [5]</td>
<td>PCTS</td>
<td>TCORW</td>
<td>0.23</td>
<td>0.38</td>
<td>0.47</td>
<td>0.60</td>
<td>0.44</td>
</tr>
<tr>
<td><strong>FluxInfer</strong></td>
<td><strong>WUDG</strong></td>
<td><strong>Weighted PageRank</strong></td>
<td><strong>0.43</strong></td>
<td><strong>0.69</strong></td>
<td><strong>0.90</strong></td>
<td><strong>0.95</strong></td>
<td><strong>0.77</strong></td>
</tr>
<tr>
<td>FluxInfer-with-CUSUM</td>
<td><strong>WUDG</strong></td>
<td><strong>Weighted PageRank</strong></td>
<td>0.23</td>
<td>0.40</td>
<td>0.62</td>
<td>0.73</td>
<td>0.53</td>
</tr>
<tr>
<td>FluxInfer-without-AD</td>
<td><strong>WUDG</strong></td>
<td><strong>Weighted PageRank</strong></td>
<td>0.13</td>
<td>0.20</td>
<td>0.32</td>
<td>0.50</td>
<td>0.38</td>
</tr>
</tbody>
</table>
In FluxInfer-with-CUSUM, we replaced the robust anomaly detection algorithm of FluxInfer with the CUSUM algorithm.
In FluxInfer-without-AD, we removed the robust anomaly detection algorithm of FluxInfer.

All KPIs (normal KPIs and anomalous KPIs) are used to construct the dependency graph and localize the root cause.
The results of FluxInfer-with-CUSUM and FluxInfer-without-AD demonstrate the effectiveness of our robust anomaly detection design.
Further, we can see that the performance of FluxInfer-with-CUSUM is better than that of FluxInfer-without-AD, which demonstrates that anomaly detection is necessary for the design of FluxInfer.
Case study

- The diagnosis results of FluxInfer and MicroCause baseline for a CPU saturation case. The bold red font represents the root cause related KPI.

<table>
<thead>
<tr>
<th>FluxInfer</th>
<th>MicroCause</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 cpu.usage</td>
<td>1 mysql.qps</td>
</tr>
<tr>
<td>2 cpu.usage_percent</td>
<td>2 mysql.innodb_rows_read</td>
</tr>
<tr>
<td>3 mysql.mem_used</td>
<td>3 mysql.active_session</td>
</tr>
<tr>
<td>4 docker.net_recv</td>
<td>4 docker.io_write</td>
</tr>
<tr>
<td>5 docker.net_send</td>
<td>5 mem.usage_percent</td>
</tr>
<tr>
<td>6 docker.io_write</td>
<td>6 mysql.io_bytes_write</td>
</tr>
<tr>
<td>7 mysql.io_bytes_write</td>
<td>7 cpu.usage</td>
</tr>
<tr>
<td>8 mem.usage_percent</td>
<td>8 mysql.bytes_sent</td>
</tr>
</tbody>
</table>

...
Case study

• The diagnosis results of FluxInfer and MicroCause baseline for a CPU saturation case. The bold red font represents the root cause related KPI.

• The diagnosis result of FluxInfer shows that the root cause related KPIs are ranked top 1 and top 2, respectively.
• However, the root cause related KPI is ranked top 7 in the diagnosis result of MicroCause baseline.
• The other eight baselines even cannot rank the root cause related KPIs in top 10.
Conclusion

• This paper presents FluxInfer, which can automatically and accurately localize the root cause related KPIs of online database performance anomalies.

• We propose an algorithm to automatically construct a Weighted Undirected Dependency Graph (WUDG) to accurately represent the dependency relationships of anomalous KPIs.

• We propose to use a weighted PageRank algorithm to traverse WUDG, which can accurately localize root cause related KPIs.

• Detailed evaluation experiments on our testbed show that the AC@3, AC@5, and Avg@5 of FluxInfer are 0.90, 0.95, and 0.77, outperforming nine baselines by 64%, 60%, and 53% on average, respectively.
Thank you!

Q&A

liuping15@mails.tsinghua.edu.cn

IPCCC 2020