PinSQL: Pinpoint Root Cause SQLs to Resolve Performance Issues in Cloud Databases

Xiaoze Liu†‡, Zheng Yin§ , Chao Zhao†‡, Congcong Ge†‡, Lu Chen† , Yunjun Gao† ,
Dimeng Li§ , Ziting Wang§ , Gaozhong Liang§ , Jian Tan§ , Feifei Li§

†*College of Computer Science, Zhejiang University, China*

‡*Alibaba-Zhejiang University Joint Institute of Frontier Technologies, China*

§ *Alibaba Group, China*

† {xiaoze, yuhao.zhao, gcc, luchen, gaoyj}@zju.edu.cn

§{yinzheng.yz, lidimeng.ldm, zizhou.wzt, gaozhong.lgz, j.tan, lifeifei}@alibaba-inc.com

Outline

- **Motivation**
- Related Work
- Problem Statement
- Methods
- Experimental Evaluation
- Conclusions

Motivation (Cont.)

Deploying database services on cloud systems has become a common practice in the industry.

……

Database problems cause 70% of application performance issues in production.

- Root Cause Analysis (RCA) of performance anomalies on cloud databases aims to diagnose the root cause of detected anomalies.

Motivation (Cont.)

Root Cause SQLs (R-SQLs) are the keys to resolve anomalies.

a) business scenario change (QPS sudden increase)

b) poor SQL statements.

c) MDL locks/Row locks.

Anomaly propagation chain: R-SQLs→H-SQLs→active session

a) A group of anomalous SQLs (R-SQLs) appear.

b) R-SQLs cause High-impact SQLs (H-SQLs) to appear simultaneously.

c) H-SQLs directly affect the instance performance, incurring the anomalies of active session.

Challenges on identifying R-SQLs of anomalies.

- Modeling the active session metric of SQLs.
- Modeling the impact of SQLs to locate H-SQLs.
- Distinguish the R-SQLs through the located H-SQLs.

PinSQL : Pinpoint Root Cause SQLs to Resolve Performance Issues in Cloud Databases

Outline

- Motivation
- **Related Work**
- Problem Statement
- Methods
- Experimental Evaluation
- Conclusions

Related Work

- Database Diagnostics Systems
  - Improved the performance of DBMS by optimizing the system.
  - Drawbacks: focus on the overall performance rather than handling anomalies.
- Classification-based RCA Methods
  - Divide the causes into a limited collection of types.
  - Drawbacks: cannot distinguish root cause SQLs.
- Top-SQL based Methods
  - Provided by large cloud vendors. Sort SQLs by metrics.
  - Drawbacks: cannot handle complex business scenarios.
- Autoregressive-based Methods
  - Predict causal dependencies in multivariate time-series data.
  - Drawbacks: large function space to search.

Outline

- Motivation
- Related Work
- **Problem Statement**
- Methods
- Experimental Evaluation
- Conclusions

Problem Statement

- ***Time-series Data*** $X = \{x_1, x_2, ..., x_N\}, x_i\ (1 \leq i \leq N) \in \mathbb{R}$
  - $x_i\ (1 \leq i \leq N)$ is an observation value at timestamp t_i
  - X is an process observation during the time period $[t_1, t_N]$ with a fixed time interval $\frac{t_N - t_1}{N}$
  - We usually use 1 second or 1 minute as the time interval to record or synchronize the time-series data.

- ***Anomaly Case*** $\mathcal{C} = (\mathcal{M}, \mathcal{Q}, a_s, a_e)$
  - $\mathcal{M}$ is the set of performance metrics, $\mathcal{Q}$ is the set of SQL templates
  - $[a_s, a_e)$ is the detected anomaly time period.

- ***Performance Metric***
  - Each Performance Metric $M \in \mathcal{M}$, indicating one specific system performance, is a time-series data sampled every second by monitoring system from the database instance during the period $[t_s, t_e)$.

Problem Statement

- ***SQL Template***
  - The SQL template (or SQL digest) replaces hard-coded values in the SQL statement with a placeholder (e.g., '?').
  - For example, a SQL template SELECT * FROM user table WHERE uid = ? includes the following SQL queries:
    - SELECT * FROM user table WHERE uid = 123456
    - SELECT * FROM user table WHERE uid = 654321
    - SELECT * FROM user table WHERE uid = 123321

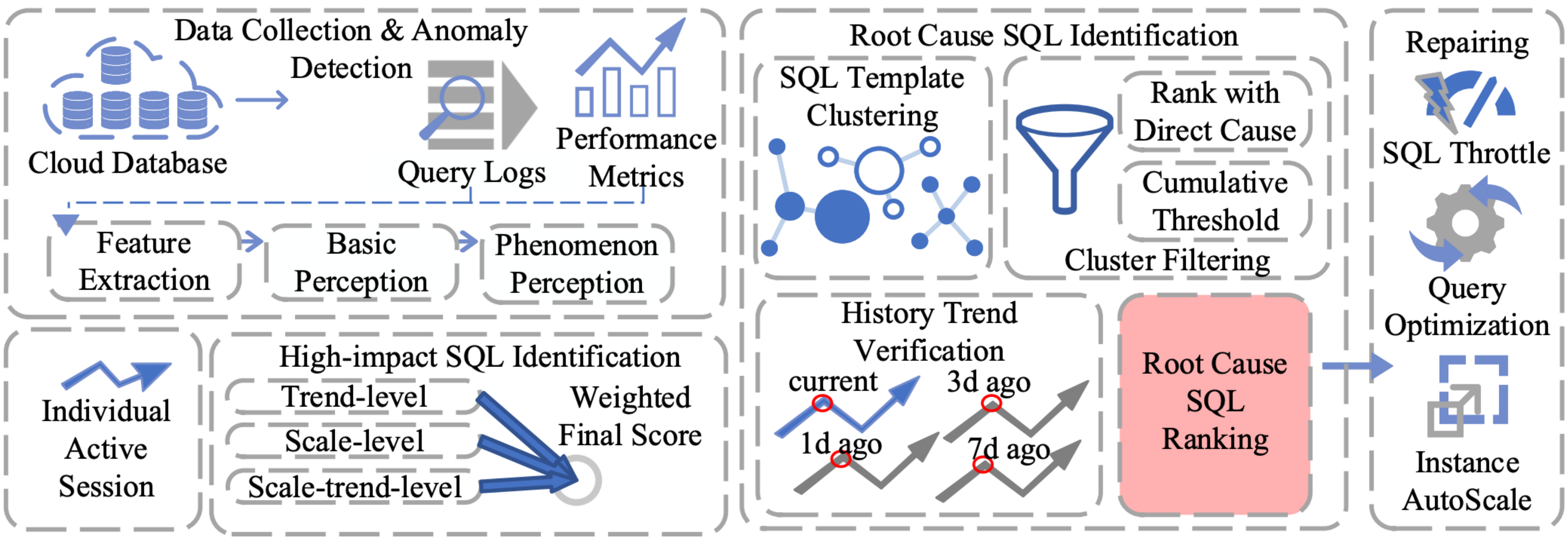
- ***Pinpointing Root Cause SQLs***
  - Given an anomaly case $\mathcal{C} = (\mathcal{M}, \mathcal{Q}, a_s, a_e)$, we aim to find a ranked list (i.e., a subset of the total SQL templates) to store R-SQLs, where higher-ranking templates are more likely to be the root causes.
  - In addition, we also aim to find another ranked list to store H-SQLs, where higher-ranking templates are more likely to be the direct causes of performance anomalies.

Outline

- Motivation
- Related Work
- Problem Statement
- **Methods**
- Experimental Evaluation
- Conclusions

Overview

- PinSQL System
  - **Data Collection & Anomaly Detection Module**
    - Collects and pre-processes the streaming raw data from millions of database instances in real-time.
  - **High-impact SQL Identification Module**
    - Triggered to locate H-SQLs hen an anomalous phenomenon is detected.
  - **Root Cause SQL Identification Module**
    - Pinpoint R-SQLs through a clustering-based strategy.
  - **Repairing Module**
    - performs various autonomous actions to handle R-SQLs.



Data Collection & Anomaly Detection

Data Collection & Pre-processing

- Collect Performance Metrics data & Query Logs data in real-time
- Aggregate SQL queries into SQL templates.

$$metric_{Q,t} = Aggregate(\{metric(q), \forall q \in Q \\ \textbf{where } t(q) \in [t, t+\Delta t)\})$$

Anomaly Detection

- Basic Perception Layer for detecting multiple anomalous features (e.g., spike up/down, levelshift, etc.).
- Phenomenon Perception Layer for detecting anomalous features of different performance metrics to recognize different anomalous phenomena.

Data Collection & Anomaly Detection

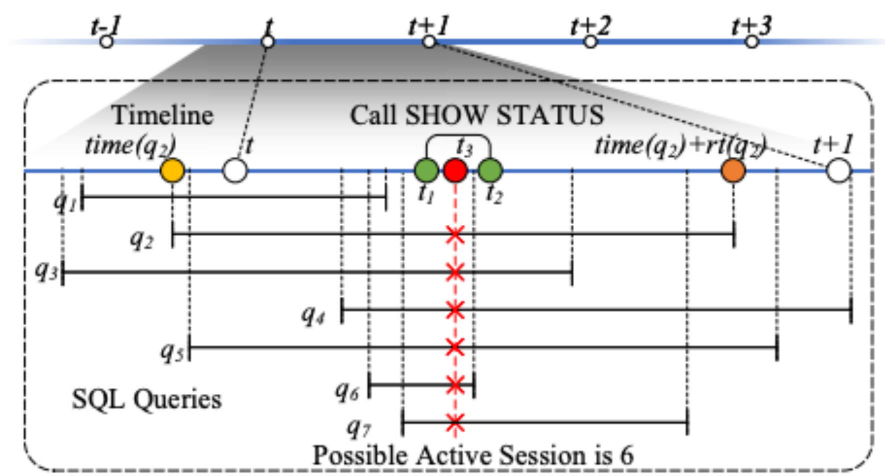
- Individual Active Session Estimation of Templates

$$P(observed(p,q)) = \frac{|p \cap [t(q), t(q) + t_{res}(q))|}{|p|}.$$

$$\mathbb{E}[session_{b_i}|\mathcal{Q})] = \sum_{Q\in\mathcal{Q}}\sum_{q\in Q} P(observed(b_i, q))$$

$$sel_t = \underset{b_i\in\{b_1,\ldots,b_K\}}{\arg\min} \left|session_t - \mathbb{E}[session_{b_i}|\mathcal{Q}]\right|$$

$$session_Q = \{\textstyle\sum_{q\in Q} P(observed(sel_t, q)), t \in \{t_s, t_s+1, \ldots, t_e\}\}$$



High-impact SQL Identification

- Correlation Coefficient

$$corr(X,Y) = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\mathbb{E}\left[(X-\mu_X)(Y-\mu_Y)\right]}{\sigma_X \sigma_Y}$$

- Weighted Correlation

$$\operatorname{cov}(X,Y;W) = \frac{\sum_i w_i \cdot (x_i - \mathrm{m}(X;W))\,(y_i - \mathrm{m}(Y;W))}{\sum_i w_i}$$

- Smooth Weight for Correlation

$$W_t = \sigma\left(\frac{t-a_s}{k_s}\right) + \sigma\left(\frac{a_e-t}{k_s}\right) - 1, t \in [t_s, t_e) \quad \text{where} \quad \sigma(x) = \frac{1}{1+e^{-x}}$$

$$\lim_{k_s \to 0} W_t = \begin{cases} 0 & \text{if } t \notin [a_s, a_e) \\ 1 & \text{otherwise} \end{cases} \text{ and } \lim_{k_s \to \infty} W_t = 1.$$

High-impact SQL Identification

- Trend-level Impact Score

$$trend(Q) = corr(session_{Qt}, session_t; W)$$

- Scale-level Impact Score

$$scale(Q) = 2 \cdot minmax_{Q \in \mathcal{Q}}\left(\sum_{t \in [a_s, a_e)} session_{Qt}\right) - 1$$

- Scale-trend-level Impact Score

$$scale\_trend(Q) = corr\left(\frac{session_{Qt}}{session_t}, session_t\right)$$

- Weighted Final Score

$$impact(Q) = \beta \cdot trend(Q) + scale\_trend(Q) + \alpha \cdot scale(Q)$$

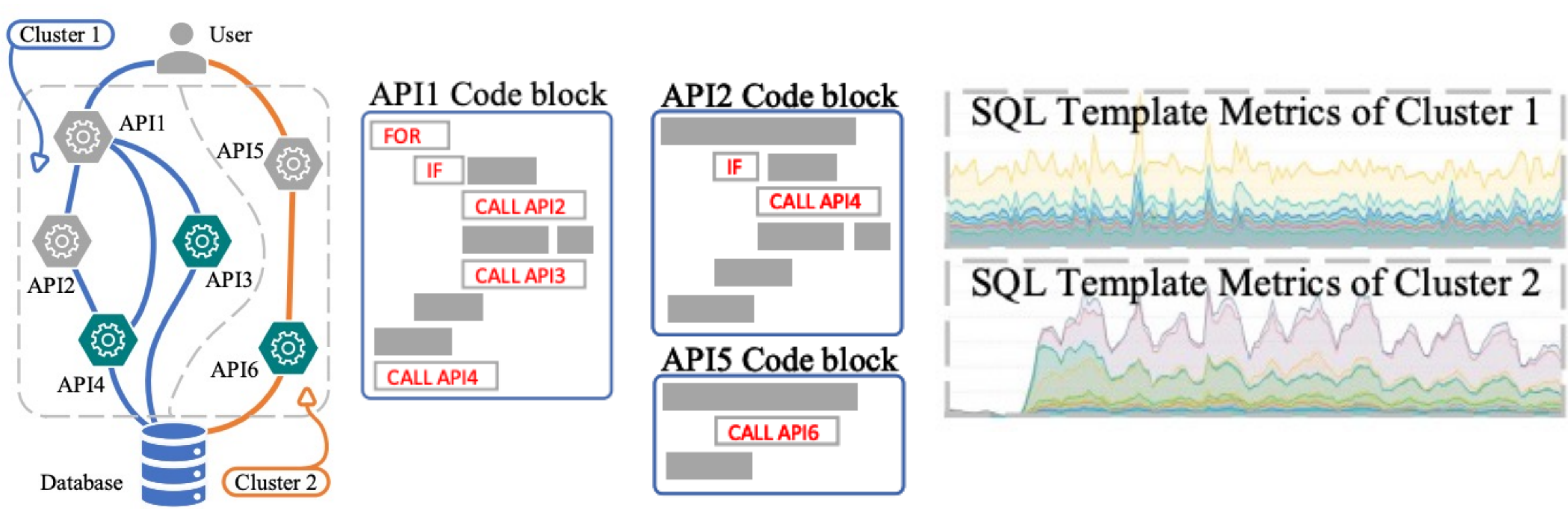
$$\text{where } \alpha = corr(session_{Q_{max}t}, session_t),\ Q_{max} = \arg\max_{Q \in \mathcal{Q}} scale(Q)$$

$$\beta = -\alpha$$

Root Cause SQL Identification

SQL Template Clustering

- To classify templates into different business logic
- Modern implementation of business logic follows the microservice architecture
- Clustering with Trend of #execution.



Root Cause SQL Identification

- **Clustering with Trend of #execution**
  - Build adjacency matrix with threshold.

$$adj_{X,Y} = corr(metric(X), metric(Y)) > \tau$$

  - Clustering with connected components of adj
- **Ranking Clusters with impact for Filtering**
  - For a cluster in the clustering result $c \in D$

$$impact(c) = \max_{Q \in c} impact(Q)$$

- **Cumulative Threshold.**
  - Iteratively calculate an accumulated active session

$$S_i = \sum_{j \in [1,i]} \sum_{Q \in D_i} session_Q$$

  - Check whether the correlation score of S_i and $session$ is larger than the threshold τ_c
  - If the threshold is reached, keep only top-i clusters.

Root Cause SQL Identification

History Trend Verification

- Verify the R-SQLs set provided by previous process.
- Observation: it is difficult for stable traffic to cause anomalies.
- Use historical data to verify whether they are R-SQLs or not.
  - Anomaly detected during the anomaly period.
  - No anomaly was detected during the relative anomaly period of N_d days ago.

Root Cause SQLs Ranking.

- Rank the remaining templates
- using the correlation between templates' number of execution and the instance active session.

Repairing Actions

Repairing Module

- Multiple configurable actions on R-SQLs and instances.

```
{
    "event_name": "SQLAvgExaminedRowsSuddenIncrease",
    "expr": "(mysql.cpu_usage.feature in [\"spike\",\"shift_up\"]
    or mysql.active_session.feature in [\"spike\", \"shift_up\"])
    and r_SQLs.examinedRows.feature in [\"spike\", \"shift_up\"]",
    "anomaly_status": "Warning",
    "suggestion": "SQL Optimization"
}
```

- SQL Throttling
  - Apply rate-limiting thresholds on R-SQLs.
- Query Optimization
  - Optimize the R-SQLs with techniques such as query rewrite & automatic indexing.
- Instance AutoScale
  - Automatically upgrade the performance configuration of the instance.

Outline

- Motivation
- Related Work
- Problem Statement
- Methods
- **Experimental Evaluation**
- Conclusions

Experimental Evaluation

- **Datasets**
  - ADAC: Anomaly cases collected from the internal database of Alibaba online services.
  - Characteristics:
    - *high database loads*
    - *complex business logic*
  - Statistics:
    - *168 anomaly cases*
    - *36 unique DB instances with 15.9 cores & 87.9GiB memory on average*
    - *1,653 minutes of time-series data*
    - *9.4 billion queries are executed.*
    - *77,450 unique SQL templates*
    - *3,357 templates on average*
- **Competitors**
  - Top-SQL-based methods that have widely applied in the industry.
- **Metrics**
  - Hits@N (N=1, 5): the percentage of correct alignment ranked at top-N.
  - MRR: the average of the reciprocal ranks of results.

Experimental Evaluation (Cont.)

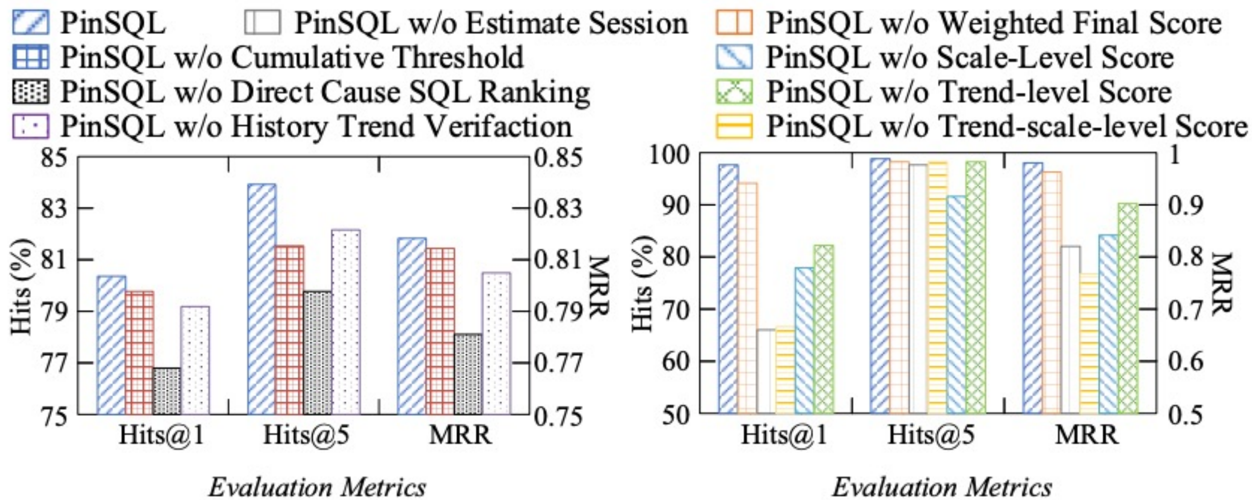
Overall results on identifying R-SQLs and H-SQLs

Method	R-SQLs				H-SQLs			
	H@1	**H@5**	**MRR**	**Time**	**H@1**	**H@5**	**MRR**	**Time**
Top-RT	31.0	56.0	0.40	2.73ms	64.3	97.0	0.75	2.73ms
Top-ER	13.7	47.6	0.28	2.27ms	44.0	67.3	0.52	2.27ms
Top-EN	6.5	6.5	0.08	2.26ms	3.0	10.7	0.08	2.26ms
Top-All	33.3	56.0	0.42	-	66.1	97.0	0.76	-
PinSQL	80.4	83.9	0.82	14.94s	97.6	98.8	0.98	8.48s

- PinSQL outperforms the Top-SQL based results on both identifying H-SQLs and R-SQLs.
- The time consumed of PinSQL is acceptable.

Experimental Evaluation (Cont.)

Ablation Study



(a) R-SQLs (b) H-SQLs

- Individual Active Session Estimation better models the performance metric.
- Considering multiple level information does capture more time-series information for identifying H-SQLs.
- Identifying H-SQLs is an essential process for pinpointing R-SQLs.

Outline

- Motivation
- Related Work
- Preliminaries
- Algorithms
- Experimental Evaluation
- **Conclusions**

Conclusions

- We develop PinSQL, an autonomous diagnosing system that includes two key features (i.e., root cause analysis and automatic repairing), to solve the problem of pinpointing root cause SQLs for the performance issues in cloud databases.

- We introduce a Data Collection And Anomaly Detection Module, which estimates the active session of each template with little performance overhead on database instances.

- We propose a High-impact SQL Identification Module, which fuses the multi-level impact of SQL templates on active session to effectively identify H-SQLs.

- We present a Root Cause SQL Identification Module, which utilizes a clustering-based strategy to select possible RSQLs via their trends on execution number. It accurately distinguishes R-SQLs based on H-SQLs by recognizing the trends of execution number.

- Comprehensive experimental results on real-world anomaly cases demonstrate the superiority of our proposed PinSQL for identifying and handling R-SQLs, compared against existing approaches.

Thank you !

Xiaoze Liu, Zheng Yin, Chao Zhao, Congcong Ge, Lu Chen, Yunjun Gao,
Dimeng Li, Ziting Wang, Gaozhong Liang, Jian Tan, Feifei Li

PinSQL: Pinpoint Root Cause SQLs to Resolve Performance Issues in Cloud Databases

ICDE 2022