DBPecker: A Graph-Based Compound Anomaly Diagnosis System for Distributed RDBMSs

Qingliu Wu*
Beijing University of Posts and Telecommunications
wql@bupt.edu.cn

Qingfeng Xiang*
Beijing University of Posts and Telecommunications
xiangqingfeng@bupt.edu.cn

Yingxia Shao†
Beijing University of Posts and Telecommunications
shaoyx@bupt.edu.cn

Qiyao Luo
Independent Researcher
sixsiebenuno@gmail.com

Quanqing Xu
Independent Researcher
xuquanqing@gmail.com

ABSTRACT

This demonstration introduces DBPecker, an integrated diagnostic platform tailored for distributed relational database systems. DBPecker leverages a graph-based anomaly modeling approach to capture inter-node dependencies and effectively localize compound anomalies, while a causality-aware metric prioritization module automatically isolates critical performance indicators. By unifying anomaly detection with a comprehensive root cause analysis pipeline, the system facilitates rapid and precise diagnosis in distributed database environments. Evaluated on a multi-node OceanBase cluster, DBPecker not only accelerates the identification of underlying anomalies but also substantially improves operational reliability, offering practical insights and actionable recommendations for real-world distributed database management.

PVLDB Reference Format:
Qingliu Wu, Qingfeng Xiang, Yingxia Shao, Qiyao Luo, and Quanqing Xu. DBPecker: A Graph-Based Compound Anomaly Diagnosis System for Distributed RDBMSs. PVLDB, 18(12): 5383 - 5386, 2025.
doi:10.14778/3750601.3750677

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Micuks/DBPecker.

1 INTRODUCTION

Accurately identifying the root cause of an incident is important to maintain the distributed database function normally in the production environment, especially since anomalies often occur in compounded ways. Compound anomalies in centralized database systems refer to coordinated combinations of multiple fundamental anomalies from OS-level or database-level that exist at the same time, combine infrastructure-level (CPU/Memory) and database-layer anomalies (excessive indexing), and create complex failure modes through multiple anomalies superimposed [2]. For instance, a database node may simultaneously experience insufficient cache capacity and I/O saturation, which together cause performance degradation through their synergistic interaction. In distributed database systems, the pattern of compound anomalies is even more complex, potentially involving multiple types of failures across various nodes [9].

Diagnosing compound anomaly in a distributed database system is not an easy task. The proliferation of distributed database system architectures has exponentially increased the complexity of distributed system observability [7]. However, current database diagnostic tools are either primarily designed for centralized databases, such as DBSherlock [8], ADDM [1], AutoMonitor [3], D-Bot [12], or only focus on a specific issue within the whole diagnostic process, such as iSQUAD [4] for slow query detection, DBCatcher [10] for anomaly detection. They exhibit significant limitations when applied to the diagnosis of distributed databases: **1) Failure of Compound Anomaly Diagnosis for Distributed Databases.** Current approaches analyze anomalies in isolation, failing to address the complexity of multiple co-occurring anomalies across different system layers (OS/database) and nodes. The simultaneous occurrence of infrastructure-level (CPU/Memory) and database-layer anomalies creates non-linear interactions that existing holistic analysis methods cannot decouple. This necessitates node-level metric correlation analysis combined with cross-node dependency tracking to identify compound failure patterns. **2) Labor-intensive Metrics Ranking of Root Cause Analysis.** While systems like DBSherlock [8] can detect anomalies, they lack automated metric prioritization mechanisms for root cause identification. The multi-node architecture of distributed databases introduces a significantly larger number of monitoring metrics compared to traditional single-node environments. Database administrators must manually inspect hundreds of correlated metrics and logs [11] through trial and error, as current approaches provide neither causal priority ranking nor interpretable metric influence quantification. This manual process leads to prolonged diagnosis times (as long as hours per incident) and inconsistent results.

To address the shortcomings of existing approaches, we argue that an effective diagnostic tool should be capable of completing the entire diagnostic pipeline, from anomaly detection to root cause analysis, and should be adaptable to distributed environments.

*Qingliu Wu and Qingfeng Xiang contributed equally to this paper.
†The corresponding author.

Proceedings of the VLDB Endowment, Vol. 18, No. 12 ISSN 2150-8097.
doi:10.14778/3750601.3750677

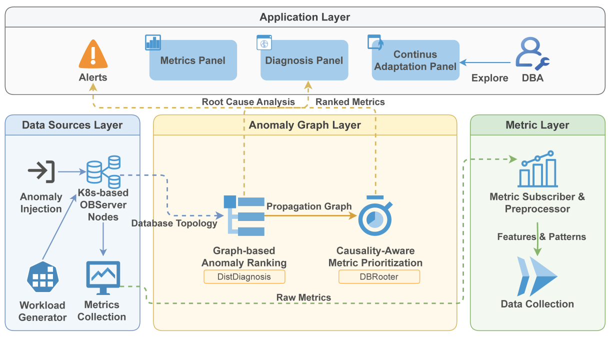


Figure 1: DBPecker system architecture for distributed database anomaly diagnosis.

In this demonstration, we introduce DBPecker, which is capable of comprehensively assisting DBAs in diagnosing distributed databases. Specifically, it possesses the following technologies:

(1) Graph-Based Compound Anomaly Localization. Our diagnosis system utilizes a graph-based anomaly modeling method named DistDiagnosis [6] that constructs compound anomaly graphs from runtime metrics across cluster nodes. Our approach achieves multi-level system observability by simultaneously capturing inter-node dependencies through graph topology. It introduces correlation-aware anomaly diagnosis using weighted PageRank to quantify node influence, systematically identifying critical fault propagation paths through the dual evaluation of topological impact and anomaly probability distributions, generating prioritized anomaly sequences.

(2) Causality-Aware Metric Prioritization. After anomaly localization (determining affected nodes and anomaly types), DBPecker performs root cause metric prioritization to help DBAs better understand the reasons behind the anomalies. In this step, we employ a causality-aware metric analysis method, referred to as DBRooter. It leverages causal graph to model the metric dependency of the current database and utilizes structural causal models to identify the most critical anomalous metrics. By leveraging DBRooter, DBAs can directly focus on the most important root cause metrics of the current database, avoiding inefficient analysis of numerous irrelevant metrics

In summary, DBPecker makes the following contributions:

- An end-to-end diagnostic framework that can effectively identify compound anomalies in distributed relational database systems.
- An efficient root cause analysis tool that assists DBAs in analyzing the vast amount of monitoring metrics.
- A suite of auxiliary components that provide data collection, anomaly simulation, and cluster metrics monitoring that helps users analyze the current operational status of the database.

2 SYSTEM ARCHITECTURE

DBPecker's architecture comprises four specialized layers. The **Anomaly Graph Layer** forms the analytical core, constructing compound anomaly graphs that model metric relationships and anomaly propagation patterns. This layer employs a distributed diagnosis engine that processes metric streams using temporal pattern recognition and feature analysis techniques. It performs DistDiagnosis to diagnose compound anomalies and leverages DBRooter to identify root cause metrics.

The **Metric Layer** handles real-time data refinement through stream processing pipelines that filter, normalize, and transform incoming metrics. This layer identifies counter metrics and converts them into differential values through window-based differentiation ($x_t - x_{t-\Delta t}$) to handle monotonically increasing counters. It incorporates window-based temporal feature extraction for pattern detection and statistical analysis, while maintaining continuous model adaptation via exponential smoothing ($\alpha = 0.2$) to accommodate evolving system behaviors.

The **Application Layer** provides operational interfaces featuring real-time cluster metric visualization, and interactive root cause analysis with metric prioritization displays. It incorporates an alert management system that triggers real-time notifications for detected fluctuations, enabling administrators to monitor system health and investigate issues through unified diagnostic dashboards.

The **Data Sources Layer** orchestrates a comprehensive data lifecycle management pipeline, initiating with distributed metric collection using SQL interfaces and monitoring tools like OBDiag, Prometheus, etc. The pipeline progresses through temporal feature extraction, identifying trend patterns, seasonal variations, and statistical outliers through window-based analysis. The layer implements automated curation of labeled datasets from both operational telemetry and injected failure scenarios, enabling incremental model updates with concept drift detection.

3 METHODOLOGY

3.1 Graph-Based Anomaly Localization

In our previous work [6], we introduce DistDiagnosis, a graph-based approach for diagnosing compound anomalies in distributed database systems. Let $G = (V, E, A, W)$ represent the compound anomaly graph where $V = \{v_i\}_{i=1}^{N}$ denotes the set of nodes corresponding to database nodes, $E \subseteq V \times V$ represents inter-node relationships, $A = \{a_i\}_{i=1}^{N}$ contains node attributes where $a_i \in [0,1]^L$ is the anomaly probability vector for node v_i, and $W = \{w_{ij}\}$ defines edge weights that captures pairwise node correlations.

In the compound anomaly graph, node attributes a_i are derived through multi-label classification using temporal metric patterns:

$$a_i^l = f_{\text{XGBoost}}(\phi(M_i^t))$$

where $M_i^t = \{m_1^{(i)}, ..., m_d^{(i)}\}$ represents the d-dimensional metric time series for node v_i over window t, ϕ denotes temporal feature extraction, and a_i^l indicates the probability of anomaly type l at node v_i. Edge weight w_{ij} is calculated by the average Pearson correlation across database nodes. For anomaly diagnosis, DistDiagnosis uses a modified PageRank algorithm that weights correlations more heavily. For node v_i, its influence score p_i is computed iteratively:

$$p_i^{(k+1)} = \frac{1-d}{N} + d \sum_{j \neq i} p_j^{(k)} \cdot (w_{ij})^2,$$

where d is the damping factor and the squared weights $(w_{ij})^2$ amplify the effect of strong correlations while diminishing weak ones.

The final anomaly score combines the influence and anomaly probabilities with power transformations to enhance differentiation:

$$s_i^l = \beta \cdot (p_i)^{\gamma_1} \cdot (a_i^l)^{\gamma_2}$$

where p_i is the node's PageRank score, a_i^l is the probability of anomaly type l at node v_i, β is a scaling factor, and γ_1 and γ_2 are power exponents. The power transformations provide better discrimination between scores, with a stronger emphasis on the anomaly probability compared to the node importance.

The final output of DBPecker is a sequence of <node, anomaly> pairs sorted by anomaly scores. This allows DBAs to identify the most impactful anomalies and gain insights into the current state of database operations. In cases where compound anomalies occur, all types of anomalies across all affected nodes will be displayed, enabling DBAs to understand all co-occurring anomalies comprehensively.

3.2 Causality-Aware Metric Prioritization

After identifying the types of anomalies that occur in the database, DBAs need to further analyze metrics to understand root causes and recovery strategies. To reduce manual analysis workload, DBPecker utilizes a metric prioritization technique, which is based on the Structural Causal Model [5]. Traditional causal approaches leverage the Dependency Graph to model the relationship across metrics. However, there are several shortcomings in Dependency Graph. First, it cannot explicitly represent the causal relationships between nodes. Second, constructing a Dependency Graph with a large number of metrics is time-consuming.

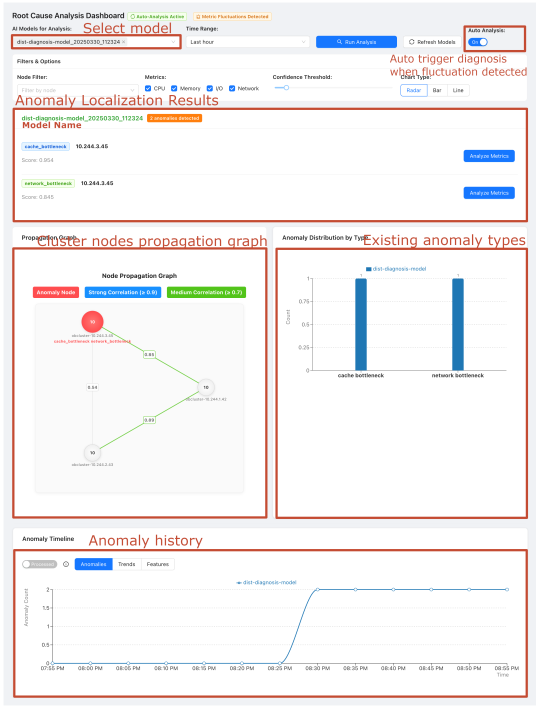
To handle the above issues, DBPecker employs a new metric prioritization approach, named DBRooter. It constructs a directed causal graph of distributed metrics. In this graph, the nodes represent specific metrics on a database node, and the directed edges represent causal relationships. Its construction process can be completed in parallel across multiple nodes. To identify root cause metrics in the distributed metric causal graph, DBPecker leverages the Structural Causal Model, which provides a formal way to model the causal dependencies between metrics, allowing for a deep understanding of how changes in one metric can affect others. Each node is associated with a structural equation that describes how the value of the node is determined by its parent nodes and possibly some exogenous (external) factors. For example, if metric Y is influenced by its parent metric X and metric Z, the structural equation of Y is written as:

$$Y = f(X, Z, \epsilon_Y),$$

where f is a regression function and ϵ_Y is an exogenous error term. Finally, DBPecker identifies the metrics with the highest anomaly scores as the root cause metrics through intervention identification.

4 DEMONSTRATION SCENARIOS

The demonstration highlights DBPecker's capability to rapidly and accurately localize compound anomalies, effectively pinpointing the metrics most closely associated with the anomalies. This facilitates a data-driven diagnosis process, empowering DBAs to efficiently target remediation efforts.



AI Models for Analysis: dist-diagnosis-model_20250330_112324
Time Range: Last hour
Auto Analysis: On
Metrics: ✓ CPU ✓ Memory ✓ I/O ✓ Network
Chart Type: Radar

dist-diagnosis-model_20250330_112324 2 anomalies detected

cache_bottleneck 10.244.3.45
Score: 0.954

network_bottleneck 10.244.3.45
Score: 0.845

Figure 2: Compound anomaly diagnosis workflow showing anomaly injection controls, real-time metric correlations, and root cause ranking visualization.

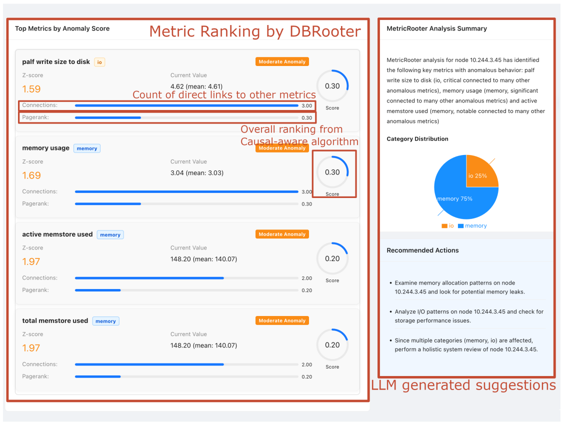
DBPecker is deployed on a three-node OceanBase(Community Edition 4.2.5.1) test cluster. Benchmark workloads are generated using industry-standard tools such as TPCC and Sysbench, thereby replicating the dynamic load characteristics observed in real-world distributed environments. Concurrently, system-level and database-specific anomalies are methodically introduced via fault injection tools such as Chaos Mesh.

4.1 Demonstration Scenario 1: Compound Anomaly Localization

This demonstration showcases DBPecker's diagnostic capabilities for compound anomalies in distributed database environments. In this scenario, demo attendees can select specific types of compound anomalies occurring in the database and observe DBPecker's anomaly diagnosis results.

As shown in Figure 2, DBPecker accelerates the diagnostic process by automatically distinguishing between primary anomaly sources and secondary symptom nodes through its weighted PageRank analysis. This capability enables database administrators to quickly understand anomaly patterns and recognize abnormal nodes.

This demonstration emphasizes how DBPecker's graph-based approach handles the complexity of distributed systems where symptoms often manifest differently than causes, significantly reducing the time required for anomaly identification.


Metric Ranking by DBRooter
Count of direct links to other metrics
Overall ranking from Causal-aware algorithm
LLM generated suggestions

Top Metrics by Anomaly Score

palf write size to disk io — Moderate Anomaly
Z-score 1.59 — Current Value 4.62 (mean: 4.61) — 0.30 Score
Connections: 3.00
Pagerank: 0.30

memory usage memory — Moderate Anomaly
Z-score 1.69 — Current Value 3.04 (mean: 3.03) — 0.30 Score
Connections: 3.00
Pagerank: 0.30

active memstore used memory — Moderate Anomaly
Z-score 1.97 — Current Value 148.20 (mean: 140.07) — 0.20 Score
Connections: 2.00
Pagerank: 0.20

total memstore used memory — Moderate Anomaly
Z-score 1.97 — Current Value 148.20 (mean: 140.07) — 0.20 Score
Connections: 2.00
Pagerank: 0.20

MetricRooter Analysis Summary

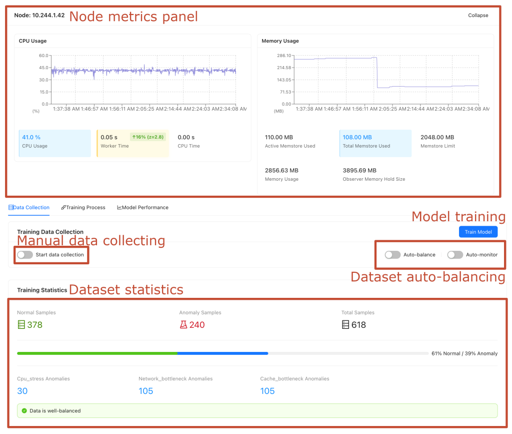
MetricRooter analysis for node 10.244.3.45 has identified the following key metrics with anomalous behavior: palf write size to disk (io, critical connected to many other anomalous metrics), memory usage (memory, significant connected to many other anomalous metrics) and active memstore used (memory, notable connected to many other anomalous metrics)

Category Distribution: io 25%, memory 75%

Recommended Actions

- Examine memory allocation patterns on node 10.244.3.45 and look for potential memory leaks.
- Analyze I/O patterns on node 10.244.3.45 and check for storage performance issues.
- Since multiple categories (memory, io) are affected, perform a holistic system review of node 10.244.3.45.

Figure 3: Metric prioritization dashboard displaying temporal outlier scores, causal influence networks, and permutation importance rankings.



Node: 10.244.1.42

41.0 % CPU Usage — 0.05 s Worker Time ↑16% (z=2.8) — 0.00 s CPU Time

110.00 MB Active Memstore Used — 108.00 MB Total Memstore Used — 2048.00 MB Memstore Limit

2856.63 MB Memory Usage — 3895.69 MB Observer Memory Hold Size

Training Data Collection: Start data collection — Auto-balance — Auto-monitor

Training Statistics

Normal Samples 378 — Anomaly Samples 240 — Total Samples 618

61% Normal / 39% Anomaly

Cpu_stress Anomalies 30 — Network_bottleneck Anomalies 105 — Cache_bottleneck Anomalies 105

Data is well-balanced

Figure 4: The 'Node metrics panel' displays real-time resource usage. The lower section shows tools for the pipeline, including data collection, training, and viewing statistics.

4.2 Demonstration Scenario 2: Metric Prioritization

This demonstration showcases DBPecker's metric analysis capabilities through its DBRooter component. After identifying the affected nodes and anomaly types, we can interact with the "Metric Analysis" module to investigate the specific metrics contributing to the observed anomalies.

As shown in Figure 3, the demonstration interface contains two primary sections: metric ranking by DBRooter and LLM-generated suggestions. This summary articulates the relationships between high-ranking metrics and provides actionable optimization recommendations tailored to the specific situation. Furthermore, as shown in Figure 4, DBPecker will display the critical metric change curves on the monitoring dashboard to further assist DBAs in root cause analysis. This demonstration highlights DBPecker's ability to bridge the gap between anomaly detection and practical resolution steps, significantly reducing the time required to repair distributed database incidents.

4.3 Demonstration Scenario 3: Auxility Functions

This demonstration illustrates the system's auxiliary capabilities, which integrate a comprehensive performance monitoring framework with systematic data collection processes. The platform consolidates metrics from both cluster-level and node-specific observations, enabling rigorous analysis of operational trends and precise performance diagnostics.

In addition, the system encompasses functionalities for training data curation and dataset management, ensuring the balanced collection of operational and anomaly datasets for continuous calibration of diagnostic methodologies. By fostering dataset management, these auxiliary functions enhance the overall resilience and strategic management of distributed database systems.

5 CONCLUSION

This paper introduces DBPecker, an end-to-end distributed database diagnosis platform. It effectively identifies compound anomalies, provides analysis of root cause metrics, and offers comprehensive diagnostic management features. This demonstration highlights DBPecker's effectiveness in supporting anomaly diagnosis in distributed databases.

ACKNOWLEDGMENTS

This work is supported by the National Science and Technology Major Project (No. 2022ZD0116315), National Natural Science Foundation of China (Nos. 62272054, 62192784), Beijing Nova Program (No. 20230484319), and State Key Laboratory of Multimedia Information Processing Open Fund (No. SKLMIP-KF-2025-07).

REFERENCES

[1] Karl Dias et al. 2005. Automatic Performance Diagnosis and Tuning in Oracle.. In *CIdR*. 84–94.
[2] Shiyue Huang et al. 2023. DBPA: A Benchmark for Transactional Database Performance Anomalies. *Proceedings of the ACM on Management of Data* (2023), 1–26.
[3] Lianyuan Jin et al. 2021. AI-based Database Performance Diagnosis. *Journal of Software* 32, 3 (2021).
[4] Minghua Ma et al. 2020. Diagnosing root causes of intermittent slow queries in cloud databases. *Proceedings of the VLDB Endowment* 13, 8 (2020), 1176–1189.
[5] Judea Pearl. 2009. *Causality: Models, Reasoning and Inference* (2nd ed.).
[6] Qingfeng Xiang et al. 2025. Distributed Database Diagnosis Method for Compound Anomalies. *Journal of Software* 36, 3 (2025), 1022.
[7] Zhenkun Yang et al. 2022. OceanBase: a 707 million tpmC distributed relational database system. *Proc. VLDB Endow.* 15, 12 (2022), 3385–3397.
[8] Dong Young Yoon et al. 2016. DBSherlock: A Performance Diagnostic Tool for Transactional Databases. In *Proceedings of the 2016 International Conference on Management of Data*. 1599–1614.
[9] Chunxi Zhang et al. 2023. Scalable and quantitative contention generation for performance evaluation on OLTP databases. *Frontiers of Computer Science* 17, 2 (2023), 172202.
[10] Guangyu Zhang et al. 2023. Dbcatcher: A cloud database online anomaly detection system based on indicator correlation. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 1126–1139.
[11] Huan Zhou et al. 2023. Scalable and adaptive log manager in distributed systems. *Frontiers of Computer Science* 17, 2 (2023), 172205.
[12] Xuanhe Zhou et al. 2024. D-Bot: Database Diagnosis System using Large Language Models. *Proceedings of the VLDB Endowment* 17, 10 (2024), 2514–2527.