International Journal of Software and Informatics, ISSN 1673-7288
http://www.ijsi.org, ijsi@iscas.ac.cn, +86-10-62661048
IJSI, 2025, 15(1): 115–137, doi: 10.21655/ijsi.1673-7288.00348

International Journal of Software and Informatics

Research Article

Distributed Database Diagnosis Method for Compound Anomalies

Qingfeng Xiang (向清风)[1], Yingxia Shao (邵鉴侠)[1], Quanqing Xu (徐泉清)[2], Chuanhui Yang (杨传辉)[2]

[1] (School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing 100876, China)

[2] (Beijing University of Posts and Telecommunications, Beijing 100876, China)

Corresponding author: Yingxia Shao, shaoyx@bupt.edu.cn

Abstract Databases are foundational components in computer services. However, performance anomalies can damage service quality. How to diagnose performance anomalies in databases has become a hot problem in industry and academia. Recently, a series of automated anomaly diagnosis methods have been proposed. They analyze the runtime status of the database and find the most likely anomalies. However, with the expansion of data scale, distributed databases are becoming increasingly popular in enterprises. In a distributed database, which is composed of multiple nodes, existing anomaly diagnosis methods struggle to effectively locate anomalies that can occur on nodes, and fail to identify compound anomalies across multiple nodes, resulting in insufficient diagnostic capabilities. To address these challenges, we propose an anomaly diagnosis method for compound anomalies in distributed databases, DistDiagnosis. It models the anomalous state of distributed databases by using a compound anomaly graph, which not only represents anomalies at each node but also captures the correlations between nodes. DistDiagnosis introduces a correlation-aware root cause ranking method, locating root cause anomalies based on the relation between nodes. In this work, we construct anomaly testing cases for different scenarios on the domestically developed distributed database OceanBase. The experimental results show that DistDiagnosis outperforms other SOTA baselines, achieving the *AC*@1, *AC*@3, and *AC*@5 values of 0.97, 0.98, and 0.98, respectively. Compared to the second-best method, DistDiagnosis improves accuracy by up to 5.20%, 5.45%, and 4.46%, respectively.

Keywords distributed database; anomaly diagnosis; root cause analysis

Citation Xiang QF, Shao YX, Xu QQ, Yang CH. Distributed database diagnosis method for compound anomalies, *International Journal of Software and Informatics*, 2025, 15(1): 115–137. http://www.ijsi.org/1673-7288/348.htm

A database management system is an important basic component of computer services, which plays a key role in Internet, finance, government units, scientific research institutions,

This is the English version of the Chinese article “面向复合异常的分布式数据库异常诊断方法. 软件学报, 2025, 36(3): 1022–1039. doi: 10.13328/j.cnki. jos.007282”

Funding items: National Natural Science Foundation of China (62272054, 62192784); National Science and Technology Major Project (2022ZD0116315); Beijing Nova Program (20230484319); Xiaomi Young Talents Program

Received 2024-05-27; Revised 2024-07-16, 2024-08-19; Accepted 2024-08-29; IJSI published online 2025-03-31

and other social sectors. In the actual use of databases, sudden performance decline may occur due to complex factors such as changes in query patterns and inefficient table schema, which results in performance anomalies. At this point, the service quality of the database system will be seriously damaged, and downstream business will be affected. Thus, it is necessary to diagnose the root causes of anomalies in time and effectively upon the occurrence of database performance anomalies. Since the traditional database anomaly diagnosis process relies heavily on the expert knowledge of the database administrator, the effectiveness of diagnosis is difficult to be guaranteed.

In recent years, to avoid the tedious manual diagnosis process, a series of automated anomaly diagnosis methods have been proposed[1]. Benefiting from the powerful ability of machine learning, these diagnosis methods can automatically analyze the performance metrics of databases, judge their running state, and detect the types of anomalies. For example, the diagnosis methods such as DBSherlock[2] and AutoMonitor[3] diagnose anomalies by analyzing the performance metrics of centralized databases, provide the overall running state of the databases, and determine the types of anomalies.

However, with the increasing complexity of the software systems and the expansion of data scale, single-machine databases cannot meet the growing business demand. Distributed databases possess data processing ability far beyond that of single-machine databases by managing multiple servers at the same time and take into account the balance between performance and fault tolerance, becoming a popular choice in industry[4]. However, due to the complexity of managing distributed database clusters, the identification and diagnosis of performance anomalies become increasingly challenging. The traditional anomaly diagnosis methods for single-machine databases face the following problems in distributed scenarios.

(1) How to accurately locate anomalies in distributed databases at the node level. A distributed database is composed of multiple nodes, and user queries are executed by the cooperation of different nodes. Overall performance anomalies of distributed databases may be caused by single-node anomalies or multi-node anomalies[5]. The existing anomaly diagnosis technology mainly makes analysis on the database level. It focuses on how to judge the overall running state and anomaly type of databases, but fails to locate the performance anomalies of nodes effectively. For example, AutoMonitor can only determine the anomaly type of a database as a whole according to the change characteristics of the performance metrics. Depending on different user queries and corresponding distributed execution plans, complex correlations exist among database nodes during operation. The existing diagnosis methods cannot perceive the interaction between nodes, and there is no effective approach to locate anomalies at the node level.

(2) How to conduct a root cause analysis of compound anomalies in distributed databases. When performance anomalies occur, the reasons for the anomalies may not be single, and the anomalies may be composed of multiple specific types of anomalies[6]. For example, when the throughput of a database decreases, a central processing unit (CPU) bottleneck and an Input/Output (IO) bottleneck may occur at the same time. The existing database diagnosis technology cannot well diagnose compound anomalies, showing a low accuracy. Additionally, in distributed databases, compound anomalies are more complicated, which may be composed of multiple types of anomalies produced at multiple nodes. The specific definitions and examples of compound anomalies can be found in Section 2.2. The existing methods do not consider how to diagnose compound anomalies at the node level when dealing with compound anomalies. For instance, DBSherlock does not consider the distribution of specific anomalies on database nodes when diagnosing compound anomalies, thus failing to effectively handle compound

anomalies in a distributed environment.

To address the above problems, an anomaly diagnosis method for compound anomalies in distributed databases is proposed in this paper, named DistDiagnosis, which realizes the root cause analysis of anomalies at the node level. Firstly, a graph-based anomaly modeling method is proposed for distributed databases, and a compound anomaly graph is constructed to abstract the running state of the distributed database, which effectively captures the relationship between nodes of the distributed database. For each distributed node, a multi-label classification technology is used to identify the anomalous states of single nodes. Secondly, a correlation-aware root cause ranking method is developed. It comprehensively utilizes the anomaly types of single nodes and the correlation between a node and the other nodes to rank all possible anomalies and generate a root cause sequence composed of $\langle node, anomaly \rangle$ pairs, so as to realize effective diagnosis of compound anomalies on nodes. In summary, DistDiagnosis can consider the anomalies on nodes and the relationship between nodes in anomaly diagnosis, which has a higher diagnostic accuracy than traditional diagnosis methods.

The main contributions of this paper are summarized as follows.

(1) A graph-based anomaly modeling method for distributed databases is proposed. It constructs a compound anomaly graph according to the runtime index of each node in a distributed database, combines with a single-node anomaly diagnoser to abstract the running state of the distributed database, and captures the relationship between nodes in the database and the anomalies of individual nodes.

(2) The correlation-aware root cause ranking technology is designed. The weighted PageRank method is used to evaluate the influence of nodes in the compound anomaly graph, and the nodes that have great influence on the database are identified. The importance of nodes in the overall system anomaly is calculated by comprehensively utilizing their influence and anomaly probabilities. On this basis, the final root cause sequence is generated.

(3) With the use of the above technology, we develop an intelligent anomaly diagnosis method DistDiagnosis for distributed databases. The experimental results show that the DistDiagnosis scheme has obvious advantages in various anomaly scenarios, and the effect of DistDiagnosis in anomaly diagnosis of distributed databases exceeds those of its counterparts, with its highest $AC@1$, $AC@3$, and $AC@5$ reaching 0.82, 0.86, and 0.92, respectively.

In this paper, Section 1 introduces the related work and research status of database anomaly diagnosis. Section 2 provides the basic knowledge, including distributed databases and distributed database anomaly diagnosis studied in this paper. Section 3 outlines the proposed method DistDiagnosis. Section 4 presents the proposed graph-based anomaly modeling method for distributed databases. Section 5 describes the designed root cause ranking technology. Section 6 verifies the effectiveness of DistDiagnosis experimentally. Section 7 makes a conclusion of the paper.

1 Database Anomaly Diagnosis Related Work

At present, a series of studies have been conducted on database anomaly diagnosis, which fall into three categories according to their diagnostic criteria[1], including time-based analysis, log-based analysis, and performance metrics-based anomaly diagnosis technology. Time-based analysis focuses on analyzing the execution time indicator in the database, which cannot effectively utilize the monitoring ability of the database and is difficult to migrate to different database systems. Log-based analysis judges the occurrence of anomalies by using the event records in the logs, and its anomaly detection effect depends on the predefined detection rules

and log granularity. The anomaly diagnosis technology based on performance metrics can easily access the metric interface provided by the database without the need for database system modification. Its extra overhead is relatively low, and is considered as a general and mainstream approach. In this paper, we mainly discuss the database diagnosis work based on performance metrics, and the proposed DistDiagnosis is also a diagnostic method based on metrics.

According to the differences of technical routes, the existing database anomaly diagnosis methods based on performance metrics can be mainly divided into two categories: rule-based diagnosis technology and machine learning-based diagnosis technology.

1.1 Rule-based diagnosis technology

Yoon *et al*.[2] proposed a database performance diagnosis framework, DBSherlock, which can help users to analyze anomalies. In the use of DBSherlock, the abnormal areas in the time series of metrics need to be first selected manually. Then, according to the differences in values of metrics between normal areas and abnormal areas, DBSherlock generates a series of predicates for judging the state of the database through the causal model. In root cause analysis, DBSherlock determines the current running state of the database by judging whether the predicates are satisfied currently. Finally, DBSherlock takes the current state of the database and the satisfaction of metric predicates as a root cause analysis report to help users locate the root causes of anomalies.

ISQUAD[7] focuses on diagnosing slow queries in the system and performes root cause analysis based on the change patterns of performance metrics. First, it identified the peaks and abrupt changes of metrics with the methods of robust threshold[8] and T-test[9], so as to find out the metrics with abnormal change patterns. Then, ISQUAD identifies the metrics with high similarity with the clustering method and further analyzes them with the Bayesian case model. In the actual diagnosis, ISQUAD judges the current anomaly type of the database by the anomaly scores of the metrics and presents the users with the abnormal metrics as the root causes.

Dundjerski *et al*.[10] proposed an automated anomaly diagnosis system for Azure SQL. The system consists of three different kinds of models: generic models, categorization models, and a rule-based expert system. The generic models are used to deal with unexpected behaviors of the database, such as query timeout, database timeout, and execution exception. The generic models calculate anomaly scores according to monitoring data and compare them with the predefined benchmark value. The categorization models aim to provide users with anomaly explanations by monitoring the specific patterns of data. The expert system accurately distinguishes specific anomalies through predefined judgment statements. Through the cooperation of these different diagnosis models, the anomaly diagnosis system can finally locate and analyze root causes of different real anomalies. However, these models are difficult to be transferred to other database systems, namely that their universality is relatively weak.

1.2 Machine learning-based diagnosis technology

Jin *et al*.[3] proposed AutoMonitor, which is a lightweight automated diagnosis framework. Firstly, AutoMonitor uses an LSTM-based autoencoder[11] to detect whether there is any performance anomaly in the database at the moment. After discovering an anomaly in the database, AutoMonitor uses Kolmogorov-Smirnov test method to calculate anomaly scores for all monitoring data and forms the anomaly vector of the database. Finally, it determines the anomaly type of the database by calculating the K-nearest neighbor (KNN) between the anomaly vector and the trained anomaly vector. Experiments demonstrate that the effectiveness of root cause analysis in AutoMonitor, leveraging machine learning, outperforms that of DBSherlock.

Huang *et al*.[6] designed a benchmark test suite, DBPA, for anomaly diagnosis. In view of the lack of evaluation standards in the field of database diagnosis at present, they designed a

portable anomaly data collection framework and open-sourced the root cause analysis dataset on MySQL and PostgreSQL. In DBPA, they used XGBoost[12], random forest[13], and other machine learning models for root cause analysis. The diagnostic effects of different methods were compared. However, the dataset establishment by DBPA and its test experiments were mainly carried out in single-machine scenarios without the consideration of more complex anomalies in distributed scenarios.

Zhou *et al*.[14] proposed D-Bot, a database diagnosis method based on a large language model (LLM). They designed a tree-based LLM diagnosis process and utilized prompt engineering for guiding the model to gradually analyze database metrics. At the same time, D-Bot provides a collaborative diagnosis process. Different LLM experts are engaged to diagnose different anomalies in the database, and the overall diagnosis process is promoted through the cooperation among experts. A summary report on performance anomalies of the database is finally generated by leveraging the analysis ability of LLM. The report generated by D-Bot describes the current performance anomalies and their corresponding explanations. However, its analysis process does not take into account the multi-node characteristic of distributed databases, and the final report cannot effectively describe anomalies in distributed databases.

In summary, most of the existing anomaly diagnosis work mainly focuses on the diagnosis of single-machine databases, ignoring the diagnosis scenarios of distributed databases. In addition, the existing methods insufficiently support the diagnosis of compound anomalies with a low diagnostic accuracy.

In addition to the above-mentioned anomaly diagnosis work, several anomaly detection methods targeting distributed environments have been proposed, such as DBCatcher[5], AutoMAP[15], and Perseus[16]. However, these methods mainly focus on how to detect whether there is an anomaly in the system but not on diagnosing specific types of anomalies. For example, DBCatcher holds that there is a correlation between database nodes and selects the most likely nodes on which anomalies occurred depending on correlation changes. AutoMAP establishes a service behavior graph according to distributed micro-service nodes and adopts a random walk technology, through which AutoMAP can select nodes that are accessed more frequently and are more prone to generating anomalies. Perseus was designed in response to the “slow failure” phenomenon in large distributed storage systems. Anomalies are found by using a lightweight regression model and dynamic threshold adjustment technology. These anomaly detection tasks are different from the main research objectives of this paper: we mainly focus on analyzing the root causes of anomalies in the database and clarify anomaly types on nodes.

2 Basic Knowledge

2.1 Distributed databases

Distributed databases provide the capability to manage large-scale server clusters and support distributed data storage and processing. For example, in the OceanBase[17] distributed database, its cluster is composed of multiple peer OBServer nodes, and user tables can be stored as partition tables on each OBServer node. The queries for such data tables stored in a distributed manner can generate corresponding distributed execution plans, which are then distributed to nodes for execution. Since each node in a distributed database can undertake data processing tasks, anomalies on any node may have a negative impact on the overall execution performance of the database. Compared with traditional centralized databases, distributed databases need to face more complex and diverse performance abnormality scenarios.

According to the nature of nodes in clusters, distributed databases can be divided into two types: being constituted by peer nodes or by non-peer nodes. For example, OceanBase based on the Shared-Nothing architecture is composed of complete peer Observer nodes. Spanner[18]

cluster processes user requests through a peer Span Server; the TiDB[19] cluster can be composed of non-peer nodes such as row-based storage nodes and column-based storage nodes. In this paper, we mainly discuss the distributed databases composed of peer nodes.

2.2 Performance anomalies of distributed databases

In the existing database anomaly diagnosis work, performance anomaly scenarios of databases are generally divided into ordinary anomalies and compound anomalies[6].

Ordinary anomalies refer to situations where the entire database experiences a specific anomaly, such as CPU or I/O bottlenecks in a single-machine database.

Compound anomalies refer to performance anomalies of the database which are caused by two or more anomalies at the same time, such as concurrent CPU bottleneck and I/O bottleneck.

In single-machine database diagnosis, the input of the anomaly classifiers designed by the existing schemes is the monitoring metric of the database as a whole, and the corresponding output is the anomaly type of the whole database. For example, when the performance of a centralized database declines due to the shortage of CPU computing resources, the existing anomaly classification methods judge that the database is in the state of CPU bottleneck.

In distributed databases, performance anomaly scenarios become more complicated. As a distributed database is composed of multiple nodes and there are influence relationships and logical correlations between the nodes, performance anomalies may occur on multiple nodes of the distributed database. Figure 1 depicts the complex anomalies that may exist in distributed databases, including single-node anomalies, multi-node anomalies, and compound anomalies.

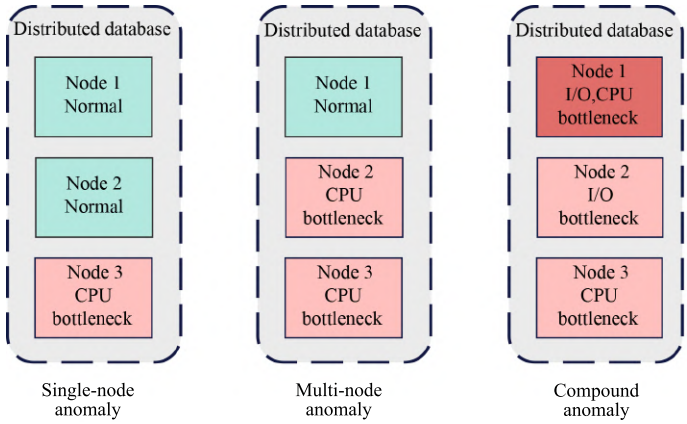

Figure 1 Complex anomaly scenarios in distributed databases

- Single-node anomaly: a specific anomaly occurs on a single node in a distributed database, e.g., the CPU bottleneck occurring on Node 3 in Figure 1.
- Multi-node anomaly: a specific kind of anomaly occurs on multiple nodes in a distributed database, e.g., the CPU bottlenecks occurring on both Nodes 2 and 3 in Figure 1.
- Compound anomaly: multiple anomalies occur on multiple nodes in a distributed database. As shown in Figure 1, both I/O and CPU bottlenecks occur on Node 1, and an I/O bottleneck and a CPU bottleneck occur on Nodes 2 and 3, respectively. This is the most complicated anomaly in a distributed database. In the example, there are two types of anomalies, I/O bottleneck and CPU bottleneck, which can be generated on different nodes of the database.

2.3 Anomaly diagnosis of distributed databases

To handle complex compound anomaly scenarios, we define the anomaly diagnosis of distributed databases as a ranking problem. For a given distributed database $DB = \{v_i\}_{i=1}^{N}$ composed of N nodes v_i, the performance metric $M_i = \{m_i^1, m_i^2, \ldots, m_i^d\}$ corresponding to each node is known, where d is the number of performance metrics, and m_i^k represents the time series formed by the k-th performance metric on node v_i, $1 \leq k \leq d$. Meanwhile, the possible L anomaly types on each node are given, with the anomaly type set being $A = \{ab_1, ab_2, \ldots, ab_L\}$.

The set $R = \{(v_i, ab_j)_m\}_{m=1}^{M}$ of all possible anomalies in the database is constituted by $\langle node, anomaly \rangle$ pairs with a total number of $M = N \times L$. In the case of performance anomalies in the database, the actual anomalies on different nodes will form a subset of R, i.e., $R^C \subseteq R$, which represents the actual root cause set. The diagnosis method needs to generate a Top-k sequence $D = ((v_i, ab_j)_n)_{n=1}^{k}$ composed of $\langle node, anomaly \rangle$ pairs as the diagnosis result according to the performance metrics of the distributed database. Elements in the set R^C are needed to be included in the sequence D as much as possible and ranked as top as possible.

This definition of the diagnostic problem has the following advantages.

(1) It can cope with the multi-node diagnosis scenarios of distributed databases. Under the traditional definition of the problem, although the existing classification method by increasing labels can support distributed node diagnosis, the number of total classification labels increases with the number of nodes, which makes it difficult to observe and analyze anomaly types.

(2) It can diagnose complex anomalies in distributed scenarios, such as distributed anomalies on nodes and compound anomalies. The final output result is a sequence of $\langle node, anomaly \rangle$ pairs, which supports the result demonstration of anomalies on different nodes, different anomaly types, and compound anomalies. In the results output by DistDiagnosis, the anomalies judged to be of great importance will rank top. When analyzing and utilizing DistDiagnosis, database administrators can check and conduct recovery one by one in line with the ranking results of $\langle node, anomaly \rangle$ pairs.

3 Overview of Our Method

DistDiagnosis is an anomaly diagnosis method for distributed databases, which is outlined in Figure 2.

In the actual online operation of databases, a large number of anomaly detection thresholds are set based on experience or business requirements. Once a performance anomaly is found in the database, a database alarm is generated. Then, DistDiagnosis first analyzes the performance metrics of each node of the database and then adopts the graph-based anomaly modeling technology for distributed databases to construct a compound anomaly graph describing the current anomalous state of the database, which summarizes the anomaly probability of each node and the correlations between nodes. Next, based on the compound anomaly graph, the correlation-aware root cause ranking is applied to generate a root cause sequence in the form of $\langle node, anomaly \rangle$ pairs as the diagnosis result. Finally, the database administrator performs the anomaly troubleshooting on the database nodes based on the diagnosis result provided by DistDiagnosis, fixes the problem, and restores the normal operation of the database.

In the overall diagnosis process, DistDiagnosis mainly includes two key technologies, namely graph-based anomaly modeling for distributed databases and correlation-aware root cause ranking, which are described as follows.

- Graph-based anomaly modeling for distributed databases

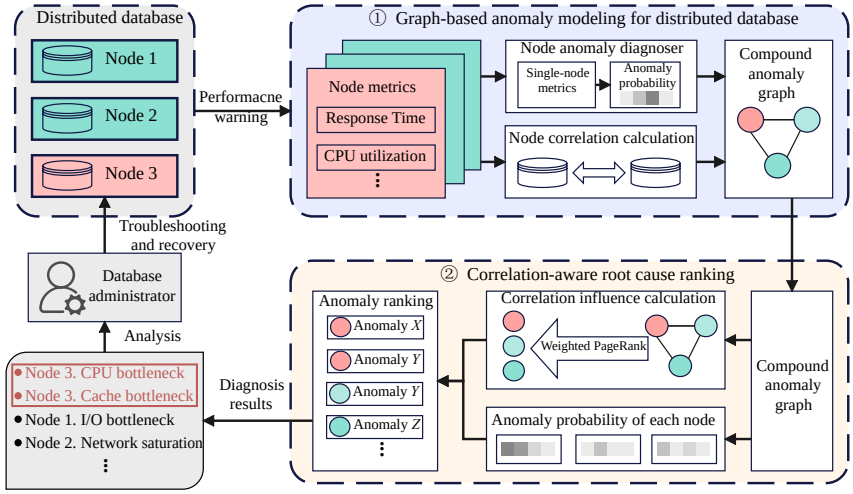

Figure 2 Structural schematic diagram of DistDiagnosis

DistDiagnosis constructs a compound anomaly graph for distributed databases. As an abstraction of the relationship between database nodes, the compound anomaly graph can capture the correlations between distributed database nodes and effectively describe the current running state of the database. When constructing the compound anomaly graph, DistDiagnosis makes each node of the database as a graph node in the compound anomaly graph. For each node, DistDiagnosis conducts anomaly diagnosis at the single-node level, judges the probability of various anomalies occurring on individual nodes, and takes the anomaly diagnosis result as the graph node attribute of the compound anomaly graph. DistDiagnosis further utilizes the performance metrics on the nodes to calculate correlations between nodes and takes the node correlations as edge weights. In the final compound anomaly graph, the graph nodes represent the anomalous state of the database nodes, and the edge weights between the graph nodes represent the influence relationships between the database nodes.

- Correlation-aware root cause ranking

In this step, DistDiagnosis evaluates the influence of each node according to the correlations between nodes and identifies the important nodes that influence the performance of the database. Based on the generated compound anomaly graph, this method uses the unsupervised graph node ranking method based on weighted PageRank[20] to calculate the influence score of each database node. Afterward, DistDiagnosis ranks $\langle node, anomaly \rangle$ pairs according to the influence scores of the nodes and the anomaly probability on each node. In the operation of a distributed database, a stronger correlation of a metric with other nodes indicates that the node has higher importance in the execution of queries, and it is more likely to become the root cause node when performance anomalies occur. The correlation-aware root cause ranking method can effectively identify such nodes and provide the final root cause sequence according to their influences.

4 Graph-based Anomaly Modeling for Distributed Databases

The expression of anomalies in traditional single-machine databases is relatively simple because only the performance anomalies at the system level need to be considered. Generally, one-dimensional anomaly vectors are enough to represent compound anomalies of the database, of which each value corresponds to the occurrence probability of an anomaly type. However, unlike single-machine scenarios, the performance anomalies in a distributed database as a

whole may be the joint result of the anomalies generated on multiple nodes. Anomaly diagnosis methods need to be able to perceive and locate anomalies at the node level. Additionally, complex relationships between nodes exist in distributed databases. Thus, the anomaly diagnosis methods need to consider the influence between nodes during diagnosis. To effectively describe the anomaly situation of nodes in distributed scenarios and capture the interaction between nodes, DistDiagnosis introduces the compound anomaly graph technology to model the anomalous state of the distributed database.

4.1 Definition of compound anomaly graph

In DistDiagnosis, a compound anomaly graph is defined as a weighted undirected graph $G = (V, E, A, W)$, where $V = \{v_i\}_{i=1}^{N}$, representing the set of graph nodes in the compound anomaly graph. For a distributed database with N nodes, the number of graph nodes in the compound anomaly graph is also N, and the graph nodes v_i correspond to the database nodes. For the sake of fully presenting the relationships of all nodes, G is a complete graph, and E contains all edges between every two nodes in V. $A = \{a_i\}_{i=1}^{N}$ represents the set of node attributes of graph nodes. The attribute a_i of each node v_i is a one-dimensional anomaly vector. The anomaly vector represents the current running state of the node, and each value corresponds to the occurrence probability of different types of anomalies. W is a set of edge weights, and its elements w_{ij} represent the correlation between nodes v_i and v_j. Larger w_{ij} indicates the higher correlation between the nodes v_i and v_j. During the construction of the compound anomaly graph, it is necessary to calculate the attributes of each node with the node-level anomaly diagnoser as well as the weights on each edge by using the node performance metrics.

In current anomaly diagnosis, graphs have been used to capture the correlations between metrics, so as to discover the metrics with the greatest influence on performance anomalies from a large number of performance metrics provided by the system. For instance, FluxInfer[21], MicroCause[22], and MonitorRank[23] abstract individual metrics into graph nodes and locate important metrics through graph node ranking. DistDiagnosis focuses on the diagnosis of anomaly types of database nodes. For the given distributed database, the compound anomaly graph G designed in this paper can effectively represent all the anomalous states of the database nodes and the relationships between nodes, realizing the effective modeling of the running state of the distributed database.

4.2 Node-level anomaly diagnoser

In the compound anomaly graph, the graph node attribute a_i represents the anomaly diagnosis result for node v_i. To support the diagnosis of compound anomalies on nodes, DistDiagnosis adopts a multi-label classification algorithm as the node-level anomaly diagnoser.

The input of the anomaly diagnoser is a multidimensional time series $M_i = \{m_i^1, m_i^2, \ldots, m_i^d\}$ composed of performance metrics on OceanBase nodes v_i. DistDiagnosis selects 40 performance metrics provided by OceanBase, such as structured query language (SQL) statistics, performance statistics, and cache, and 20 host physical resource metrics monitored by dstat, including CPU statistics, memory statistics, I/O statistics, and network statistics. The performance metrics are collected at an interval of 5 s, and the metric time series within a time window of 30 s is taken for diagnosis. In summary, for each metric time series m_i^k, $1 \leq k \leq d$, the sampling interval is 5 s, and the total time is 30 s, with $d = 60$.

For L possible anomaly types $\{ab_1, ab_2, \ldots, ab_L\}$, the output of the node anomaly diagnoser is a L-dimensional vector $a_i = (score_i^1, score_i^2, \ldots, score_i^L)$, in which each dimension of $score_i^l$ represents the probability that the node v_i generates the anomaly ab_l of the l-th type. DistDiagnosis uses XGBoost as the concrete model of the node anomaly diagnoser and trains L binary classification models for predicting probability to realize simultaneous diagnosis

of various anomaly types. To meet the input requirements of XGBoost, DistDiagnosis flattens M_i into a one-dimensional vector.

4.2.1 *Training of diagnoser*

In the training stage, the database administrator needs to assist in the anomaly diagnosis and training data annotation. When performance anomalies in the database occur, the database administrator first defines the anomaly nodes manually and annotates the specific anomaly types on the nodes. After the manual anomaly analysis, the diagnosis result of each node is added to the training data of the node-level anomaly diagnoser. As more data are added, the node-level anomaly diagnoser gradually generates more accurate prediction results and assist the database administrator in diagnosis.

4.2.2 *Prediction of graph node attributes*

When using the node-level anomaly diagnoser to construct the compound anomaly graph, the diagnoser generates the single-node anomaly vector for the database node as the node attribute in the compound anomaly graph. The advantage of using the node-level anomaly diagnoser is that it avoids the accuracy decline caused by high-dimensional inputs. In the use of the compound anomaly graph technology, DistDiagnosis decomposes the complex multi-node classification task into multiple single-node classification tasks. Because of no need to train a complex diagnosis model to predict all nodes simultaneously, better diagnosis results are achieved. In addition, distributed databases generally have the ability to expand or shrink the capacity of nodes dynamically, allowing for the addition or removal of running nodes based on containerized environments. Using a node-level anomaly diagnoser can better adapt to the changes in the number of nodes in the clusters and improve the scalability of DistDiagnosis.

4.3 Node correlation calculation

In the compound anomaly graph, DistDiagnosis measures the correlations between nodes through Pearson correlation coefficients[24], which can effectively calculate the influence relationships between time series. For two time series x and y, their cross-correlation coefficient is calculated as follows:

$$r(x,y)=\frac{\sum(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum(x_i-\bar{x})^2\sum(y_i-\bar{y})^2}} \tag{1}$$

The corresponding performance metrics of the given nodes v_i and v_j are $M_i=\{m_i^1, m_i^2,\ldots,m_i^d\}$ and $M_j=\{m_j^1,m_j^2,\ldots,m_j^d\}$, respectively. These performance metrics fully reflect the current status of nodes and can be used to measure the running performance of nodes. DistDiagnosis uses the correlations of performance metrics between nodes to represent the correlations of execution performance and processing status between nodes. At the same time, to comprehensively consider different metrics on database nodes, DistDiagnosis calculates the average of the cross-correlation coefficients of each metric, which is taken as an edge weight between two nodes, with calculation method shown below:

$$w_{ij}=\frac{1}{d}\sum_{k=1}^{d}|r(m_i^k,m_j^k)| \tag{2}$$

In the final compound anomaly graph, the edge weights between graph nodes represent the influences between nodes of the distributed database. For the two nodes with a higher weight, the correlation between performance metrics is higher during their running, which correspondingly reflects the stronger correlation of performance during the operation of the database.

5 Correlation-aware Root Cause Ranking

In the root cause analysis of anomalies, a simple way is to rank the $\langle node, anomaly \rangle$ pairs directly according to the prediction probability of each anomaly. However, this approach has certain shortcomings. It treats all nodes equally, completely depends on the anomaly detection ability on the single-node level, and fails to consider the complex relationships between nodes in the distributed database. In the distributed execution of user queries, some nodes have a greater influence on the overall execution performance of database due to reasons such as data skew[25], uneven division of partitioning keys[26], and execution plan operator placement[27]. To effectively consider the correlations between nodes, we design the correlation-aware root cause ranking technology. DistDiagnosis judges the influence of a node in the overall database according to the correlations of metrics between nodes and carries out the final root cause ranking by combining the influence and anomaly probability of each node.

5.1 Node correlation influence calculation

Firstly, the influence score of each node of the distributed database is calculated. In the compound anomaly graph, each graph node corresponds to a node of the distributed database, and an edge weight between nodes reflects the correlation of performance between two nodes. By analyzing the influence of each graph node in the compound anomaly graph, the influence and importance of each database node in the whole database can be obtained.

To effectively consider the whole compound anomaly graph when calculating the influence of graph nodes, DistDiagnosis uses the weighted PageRank algorithm to calculate the influence of each node. This algorithm is an unsupervised node ranking method on a weighted graph. Different from the ordinary PageRank method, it can consider the differences caused by different weights on each edge at the same time in the evaluation of graph nodes based on random walk. For a node with a higher weight on an adjacent edge, the weighted PageRank algorithm yields a greater influence and assigns a higher ranking to it. For each node v_i, its influence calculated by the weighted PageRank algorithm is as follows:

$$PR(v_i) = (1-d) + d \times \sum_{v_j \in N(v_i)} PR(v_j) \times w_{ij} \times w_{ji} \tag{3}$$

where $N(v_i)$ represent all the neighboring nodes of v_i; d is a hyperparameter; w_{ij} is the outgoing edge weight of v_i; and w_{ji} is the incoming edge weight of v_i.

Since the compound anomaly graph is an undirected graph, $PR(v_i)$ reduces to

$$PR(v_i) = (1-d) + d \times \sum_{v_j \in N(v_i)} PR(v_j) \times w_{ij}^2 \tag{4}$$

In each round of calculation of the influence score $PR(v_i)$, the edge weights of the node v_i and all its neighbors are aggregated and reflected in the final value of influence. For the nodes with higher correlations with other database nodes, they have higher adjacent edge weights in the compound anomaly graph. After the calculation by weighted PageRank, such important nodes also get higher influence scores. The nodes with higher influence in the compound anomaly graph have greater influences on database performance and are prone to becoming true root cause nodes.

5.2 Root cause ranking

In the final root cause ranking, DistDiagnosis considers both the influence of each node and its anomaly probability. In the ranking, the importance of each $\langle node, anomaly \rangle$ pair (v_i, ab_l)

is calculated by Eq. (5):

$$\textit{Importance}_{il} = PR(v_i) \times \textit{score}_i^l \tag{5}$$

which is the product of the influence of each node and the probability of the anomaly.

According to the importance score, DistDiagnosis ranks all possible $\langle node, anomaly \rangle$ pairs and forms a Top-k root cause diagnosis sequence $D = ((v_i, ab_j)_n)_{n=1}^k$ as the output root cause analysis result. For a distributed database $DB = \{v_i\}_{i=1}^N$ composed of N nodes, if each node may produce L kinds of anomalies, the maximum value of k is $N \times L$.

Compared with the ranking directly based on the anomaly score $\textit{score}_i^l$ of $\langle node, anomaly \rangle$ pair (v_i, ab_l), the correlation-aware root cause ranking method effectively utilizes the compound anomaly graph and also introduces the correlations between nodes through the weighted PageRank algorithm. In the execution of the distributed database, the metric with stronger correlations with other nodes represents the higher importance of the node in the query execution process, and it is more likely to become a root cause node for performance anomalies. Accordingly, its corresponding anomaly should be at the top of the final result. The correlation-aware root cause ranking method can effectively identify such nodes and correct the root cause diagnosis sequence according to their importance. The overall execution flow of DistDiagnosis is described in Algorithm 1.

Algorithm 1. DistDiagnosis anomaly diagnosis algorithm.

Input: $DB = \{v_i\}_{i=1}^N$ nodes of the current distributed base;
$M = \{M_i\}_{i=1}^N$: the performance metrics of each node;
Output: Root cause sequence D constituted by $\langle node, anomaly \rangle$ pair (v_i, ab_l).

1. //Graph-based anomaly modeling for distributed database
2. **for** $v_i \in DB$ **do**
3. Graph node attribute calculation $a_i \leftarrow$ *XGBoostClassifier*(M_i), where $a_i = (\textit{score}_i^1, \textit{score}_i^2, \ldots, \textit{score}_i^L)$;
4. **for** $v_j \in DB$ and $v_j \neq v_i$ **do**
5. Adjacent edge weight calculation for node, $w_{ij} \leftarrow (1/d) \times \sum_{k=1}^d |r(m_i^k, m_j^k)|$, where $m_i^k \in M_i$, $m_j^k \in M_j$, r is the Pearson correlation coefficient;
6. **end for**
7. **end for**
8. //Correlation-aware root cause ranking
9. **while** $PR(v_i)$ change in each round is greater than a certain threshold, **do**
10. **for** $v_i \in DB$ **do**
11. $PR(v_i) = (1-d) + d \times \sum_{v_j \in N(v_i)} PR(v_j) \times w_{ij}^2$;
12. **end for**
13. **end while**
14. **for** $v_i \in DB$ **do**
15. $\textit{Importance}_{il} \leftarrow PR(v_i) \times \textit{score}_i^l$;
16. **end for**
17. $D \leftarrow$ according to $\textit{Importance}_{il}$, all (v_i, ab_l) pairs are ranked;

6 Experimental Analysis

6.1 Experimental setting

To verify the effectiveness of DistDiagnosis in distributed database diagnosis, we conduct a comparative experiment in multiple anomaly scenarios. This section introduces the experimental settings.

6.1.1 *Experimental environment*

This experiment is conducted in a distributed cluster composed of four Linux servers, and the operating system is CentOS 7.9. The configuration of each server includes a 32-core AMD EPYC Milan@2.55 GHz processor, 64 GB of memory, and 500 GB of solid state drive (SSD). The domestic distributed database used in the experiment is OceanBase community version 4.3.0. Three servers are taken as OBServer nodes to deploy the OceanBase distributed cluster, and the remaining one server is used for pressure test.

6.1.2 *Anomaly types*

Table 1 lists the anomaly types used in this experiment. Six anomaly types are selected by referring to Refs [2, 3, 6] and considering the suggestions of OceanBase DBA.

Table 1 Common anomaly types in OceanBase

Anomaly type	Description
CPU bottleneck	Processes outside the database preempt CPU computational resources of the node
I/O bottleneck	Processes outside the database preempt I/O resources of the node
Network saturation	Processes outside the database preempt network resources of the node
Cache bottleneck	Cache size of OceanBase database node is insufficient
Heavy workload	Concurrent requests of users and transactions are too large
Too many indexes	Too many unnecessary indexes exist in the data table, affecting performance

Among these six anomaly types, the CPU bottleneck, the I/O bottleneck, the network saturation, and the cache bottleneck can be triggered on one or more nodes underlying the database. For heavy workload and too many indexes, their triggering can be reflected on all nodes at the same time. For example, since the data table is stored in all nodes of the database, the anomaly of too many indexes occurs on all nodes that store the data table when too many indexes are established on the data table.

In the collection of anomaly data, the database runs TPC-C workload first, and then different kinds of anomalies are triggered. For the CPU bottleneck and the I/O bottleneck, stress-ng is used in this experiment to preempt the corresponding resources on the corresponding node hosts. In the experiment, the cache bottleneck is triggered by adjusting the parameter memstore_limit_percentage of a single node in OceanBase. The heavy workload is triggered by sending a large number of concurrent requests. The network saturation is caused by the built-in traffic control tool tc in Linux. Before the pressure test, we establish indexes on the data table manually to trigger the anomaly of too many indexes.

6.1.3 *Anomaly scenarios*

According to the anomaly types in Table 1, we construct the following four test scenarios for anomaly diagnosis:

- Single-node anomaly scenario: one of the three nodes in the distributed database generates an anomaly in Table 1.
- Multi-node anomaly scenario: two of the three nodes in the distributed database generate the same kind of anomaly.
- Single-node compound anomaly scenario: a node of a distributed database generates different kinds of anomaly.
- Multi-node compound anomaly scenario: multiple nodes of a database generate anomalies of different kinds.

Table 2 shows all the anomaly scenarios used in the test in detail.

Table 2 Description of anomaly scenarios

Scenario	No.	Anomaly type	Scenario description
Single-node anomaly scenario	A1	CPU bottleneck	Triggering CPU bottleneck on one node
	A2	I/O bottleneck	Triggering I/O bottleneck on one node
	A3	Network saturation	Triggering network saturationon one node
	A4	Cache bottleneck	Triggering cache bottleneck on one node
Multi-node anomaly scenario	B1	CPU bottleneck	Triggering CPU bottlenecks on two nodes
	B2	I/O bottleneck	Triggering I/O bottlenecks on two nodes
	B3	Network saturation	Triggering network saturation on two nodes
	B4	Cache bottleneck	Triggering cache bottlenecks on two nodes
Single-node compound anomaly scenario	C1	CPU bottleneck + I/O bottleneck	Triggering CPU bottleneck and I/O bottleneck on one node
	C2	CPU bottleneck + network saturation	Triggering CPU bottleneck and network saturation on one node
	C3	I/O bottleneck + network saturation	Triggering I/O bottleneck and network saturation on one node
	C4	Cache bottleneck + I/O bottleneck	Triggering cache bottleneck and I/O bottleneck on one node
Multi-node compound anomaly scenario	W1	CPU bottleneck + heavy workload	Triggering CPU bottleneck on one node and heavy workload in database
	W2	I/O bottleneck + heavy workload	Triggering I/O bottleneck on one node and heavy workload in database
	W3	Network saturation + too many indexes	Triggering network saturation on one node and too many indexes in database
	W4	Cache bottleneck + too many indexes	Triggering cache bottleneck on one node and too many indexes in database

6.1.4 *Metric collection*

The existing mainstream distributed databases all provide a comprehensive monitoring system, which can detect various metrics of the whole system and the distributed nodes, including detailed data of the operating system, transaction, disk, CPU, and other aspects. In the OceanBase distributed database, there are more than 400 performance metrics available on each node. However, some of these metrics are weakly related to database performance, providing little help for database diagnosis. In this experiment, we manually select 40 key performance metrics and also adopt 20 host performance metrics of CPU, memory, disks, and network statistics which are provided by dstat, a Linux tool. During the collection of experimental data, a stress test is conducted on the database to simulate its normal operation scenario, and the metrics of each node of OceanBase are detected every 5 s, which are then stored in the form of time series. For each specific anomaly scenario, 10 groups of data are collected by changing anomaly nodes or changing the degree of anomaly triggering. Consequently, a total of 160 groups of anomaly data are formed, and each group of data is 180 s long.

6.1.5 *Baselines*

In the experiment, we use the following baselines.

- DBSherlock: when diagnosing anomalies, it forms anomaly filtering predicates about key metrics according to the anomaly areas specified by users and judges the anomalous state according to the changes of metrics, which has strong interpretability.
- AutoMonitor: it diagnoses anomalies with the improved KNN algorithm, which can effectively identify the changes of database metrics and analyze the anomaly type that best matches the current running state of the database.
- XGBoost and RandomForest: referring to the baselines used in DBPA, we also use other machine learning-based classifiers in this experiment, including XGBoost and RandomForest.

For the distributed anomaly diagnosis problem studied in this paper, the diagnosis result

is a sequence of $\langle node, anomaly \rangle$ pairs. However, the existing methods are not designed for distributed anomaly diagnosis. To migrate the baselines to the distributed diagnosis scenarios, we train multiple binary classifiers for DBSherlock, XGBoost, and RandomForest to predict each $\langle node, anomaly \rangle$ pair. Then the $\langle node, anomaly \rangle$ pairs predicted by the methods are formed into a diagnosis sequence by their prediction scores from high to low. The relative rankings of those with the same prediction scores are random. The anomaly diagnosis method of AutoMonitor based on KNN can support scoring different anomaly types, thus directly enabling the generation of the anomaly diagnosis sequence.

6.1.6 *Evaluation indexes*

In this experiment, *Precision@k* and *AC@k* are used to evaluate the diagnostic performance of different methods, which are widely applied in existing work[21, 22, 28]. *Precision@k* represents the accuracy of containing correct root causes in the first k diagnosis results, while *AC@k* represents the average accuracy in multiple anomaly cases. The diagnostic accuracy of case c in a given anomaly scenario set C is

$$Precision@k = \frac{\sum_{i<k} D^c[i] \in R^c}{\min(k, |R^c|)} \tag{6}$$

For the whole anomaly scenario set C, *AC@k* is

$$AC@k = \frac{1}{|C|} \sum_{c \in C} Precision@k \tag{7}$$

where R^c is the actual root causes of the anomaly case; $|R^c|$ and $|C|$ are the numbers of elements in R^c and C, respectively; and $D^c[i]$ represents the i-th root cause in sequence D^c which is the result of the root cause analysis.

6.2 Effect comparison and analysis

6.2.1 *Single-node anomaly scenarios*

Table 3 shows the *Precision@k* values of different methods in four single-node anomaly scenarios, and Figure 3 displays the *AC@k* values of the methods in single-node anomaly scenarios.

Table 3 Comparison of *Precision@k* among different methods in single-node anomaly scenarios

Anomaly scenario	A1			A2			A3			A4		
	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$
DBSherlock	0.83	0.86	0.86	0.77	0.79	0.80	0.68	0.73	0.79	0.43	0.55	0.63
AutoMonitor	0.83	0.90	0.95	0.83	0.88	0.89	0.62	0.77	0.84	0.48	0.57	0.71
RandomForest	0.84	0.84	0.84	0.94	0.95	0.95	0.77	0.85	0.92	**0.88**	**0.91**	**0.93**
XGBoost	0.80	0.80	0.80	0.95	0.95	0.95	0.79	0.88	0.90	0.74	0.80	0.87
DistDiagnosis	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.81**	**0.94**	**0.96**	0.82	0.82	0.86

In single-node anomaly scenarios, the experimental data reflect that the diagnostic performance of DistDiagnosis is generally superior to other baselines. DistDiagnosis has certain advantages in detecting anomalies on single nodes of the distributed database and can effectively locate the anomalous node among multiple nodes. In the diagnosis of CPU bottleneck (A1) and I/O bottleneck (A2), DistDiagnosis effectively leverages node-level system performance metrics, and its diagnostic performance significantly outperforms that of other methods. Compared with the second-best method RandomForest, DistDiagnosis improves *AC@*1, *AC@*3, and *AC@*5 by 0.05, 0.05, and 0.04, respectively in single-node anomaly scenarios.

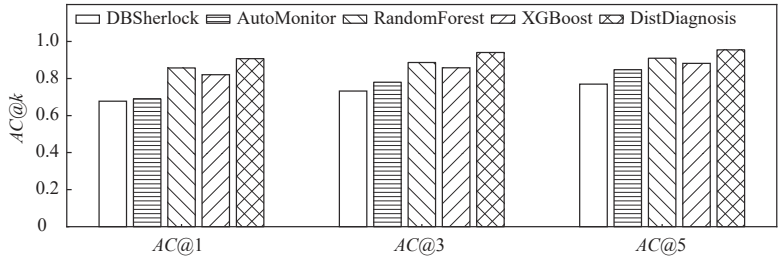

Figure 3 Comparison of $AC@k$ among different methods in single-node anomaly scenarios

6.2.2 *Multi-node anomaly scenarios*

Table 4 and Figure 4 present the $Precision@k$ and $AC@k$ of different methods in four multi-node anomaly scenarios.

Table 4 Comparison of $Precision@k$ among different methods in multi-node anomaly scenarios

Anomaly scenario	B1			B2			B3			B4		
	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$
DBSherlock	0.82	0.83	0.84	0.84	0.88	0.92	0.73	0.75	0.80	0.65	0.73	0.77
AutoMonitor	0.84	0.93	0.93	0.83	0.85	0.90	0.85	0.84	0.94	0.85	0.90	0.92
RandomForest	0.98	0.98	1.00	0.96	0.95	1.00	0.95	0.97	0.97	0.87	0.88	0.88
XGBoost	1.00	0.98	**1.00**	0.95	0.97	1.00	0.96	0.97	0.97	0.88	0.90	0.90
DistDiagnosis	**1.00**	**1.00**	0.97	**0.97**	**1.00**	**1.00**	**0.96**	**0.98**	**0.98**	**0.93**	**0.92**	**0.94**

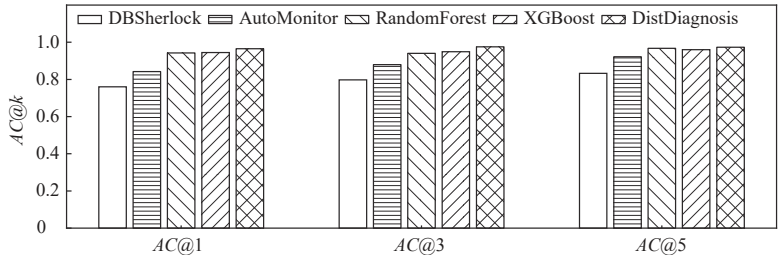

Figure 4 Comparison of $AC@k$ among different methods in multi-node anomaly scenarios

Experimental results demonstrate that DistDiagnosis can still effectively diagnose anomalies when they occur on multiple nodes in the distributed database. In these anomaly scenarios, DistDiagnosis can basically achieve optimal or near-optimal results, demonstrating strong generalizability. In regard to the average diagnostic effects of multi-node anomaly scenarios, the $AC@1$, $AC@3$, and $AC@5$ of DistDiagnosis reach 0.96, 0.98, and 0.98, respectively, which outperform all the baselines (cf. Figure 4). When calculating the compound anomaly graph, DistDiagnosis can consider the performance correlations between nodes and extract the final ranking result according to the performance correlations. Meanwhile, when diagnosing anomalies for each node, DistDiagnosis adopts an anomaly diagnoser at the node level, which can effectively avoid the noise and interference caused by metrics of other nodes when diagnosing multiple nodes at the same time, thus providing more accurate diagnosis results.

6.2.3 *Single-node compound anomaly scenarios*

Table 5 and Figure 5 show the effects of different diagnosis methods in the single-node compound anomaly scenarios. In these scenarios, the $AC@1$, $AC@3$, and $AC@5$ of DistDiagnosis can reach 0.97, 0.98, and 0.98, which are 0.03, 0.05, and 0.04 higher than those of the second-best method RandomForest, respectively. In summary, DistDiagnosis still has certain advantages in the scenarios similar to compound anomalies in traditional single-machine

databases. The proposed method has versatility in the diagnostic scenarios it can handle, making it capable of addressing various diagnostic tasks.

Table 5 Comparison of *Precision@k* among different methods in single-node compound anomaly scenarios

Anomaly scenario	C1			C2			C3			C4		
Precision@k	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$
DBSherlock	0.83	0.85	0.84	0.68	0.71	0.74	0.62	0.64	0.68	0.55	0.57	0.67
AutoMonitor	0.81	0.86	0.88	0.70	0.76	0.77	0.65	0.70	0.78	0.50	0.58	0.70
RandomForest	0.82	0.89	0.89	1.00	0.96	0.96	0.93	0.95	0.97	0.94	0.87	0.95
XGBoost	0.90	0.92	0.93	0.92	0.92	0.94	0.97	0.97	0.97	0.93	0.93	0.94
DistDiagnosis	**0.93**	**0.98**	**0.98**	**1.00**	**1.00**	**1.00**	**1.00**	**0.98**	**0.98**	**0.94**	**0.97**	**0.97**

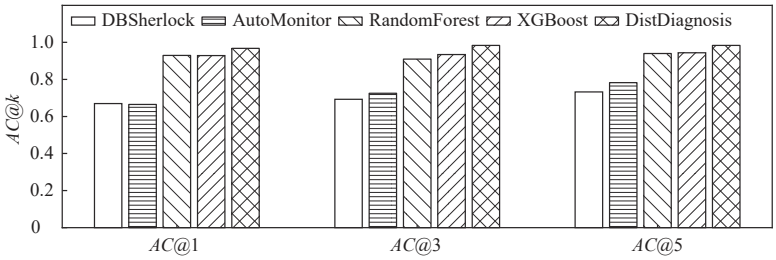

Figure 5 Comparison of *AC@k* among different methods in single-node compound anomaly scenarios

6.2.4 *Multi-node compound anomaly scenarios*

Table 6 and Figure 6 show the results of different diagnosis methods in multi-node compound anomaly scenarios. In these scenarios, DistDiagnosis has obvious advantages. The *AC@*1, *AC@*3, and *AC@*5 of DistDiagnosis can reach 0.90, 0.91, and 0.92, which are 5.20%, 5.45%, and 4.46% higher than those of XGBoost, respectively (cf. Figure 6). As for *AC@*1, which is associated with only the first anomaly diagnosis term, there is a gap of 0.05 between DistDiagnosis and the second-best method XGBoost. This shows that DistDiagnosis can still maintain its ability to accurately identify anomalies in multi-node compound anomaly scenarios and select the most likely root cause as the first ranking for users.

Table 6 Comparison of *Precision@k* among different methods in multi-node compound anomaly scenarios

Anomaly scenario	W1			W2			W3			W4		
	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$
DBSherlock	0.64	0.69	0.75	0.83	0.83	0.92	0.53	0.53	0.61	0.57	0.61	0.73
AutoMonitor	0.72	0.73	0.76	0.69	0.73	0.89	0.72	0.82	0.87	0.55	0.53	0.74
RandomForest	0.88	0.90	0.90	0.86	0.91	0.91	0.81	0.81	0.88	0.73	0.78	0.82
XGBoost	**0.93**	**0.93**	**0.95**	0.88	0.88	0.88	0.90	0.90	0.91	0.75	0.75	0.80
DistDiagnosis	0.90	0.92	0.93	**0.95**	**0.95**	**0.95**	**0.92**	**0.96**	**0.97**	**0.81**	**0.84**	**0.84**

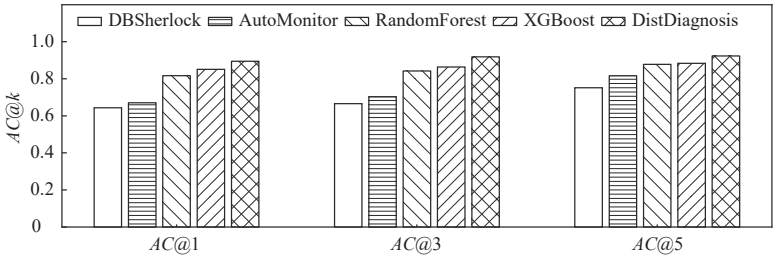

Figure 6 Comparison of *AC@k* among different methods in multi-node compound anomaly scenarios

6.3 Ablation experiment

Table 7 presents the the ablation experiment, in which DistDiagnosis-w/o-CR stands for the DistDiagnosis method without the use of the correlation-aware root cause ranking technology.

Table 7 Comparison of $AC@k$ among DistDiagnosis-w/o-CR, XGBoost, and DistDiagnosis

Anomaly scenario	Method	$AC@1$	$AC@3$	$AC@5$
Single-node anomaly scenario	XGBoost	0.82	0.86	0.88
	DistDiagnosis-w/o-CR	0.88	0.93	0.91
	DistDiagnosis	**0.91**	**0.94**	**0.95**
Multi-node anomaly scenario	XGBoost	0.95	0.95	0.97
	DistDiagnosis-w/o-CR	0.95	0.96	0.96
	DistDiagnosis	**0.97**	**0.98**	**0.97**
Single-node compound anomaly scenario	XGBoost	0.93	0.93	0.94
	DistDiagnosis-w/o-CR	0.94	0.94	0.85
	DistDiagnosis	**0.97**	**0.98**	**0.98**
Multi-node compound anomaly scenario	XGBoost	0.85	0.86	0.88
	DistDiagnosis-w/o-CR	**0.90**	0.90	0.91
	DistDiagnosis	0.89	**0.92**	**0.92**

The experimental results show the followings. 1) Although XGBoost is used as the single-node diagnoser in the proposed method, DistDiagnosis and DistDiagnosis-w/o-CR present better results than XGBoost. This is because of the proposed anomaly modeling technology. DistDiagnosis and DistDiagnosis-w/o-CR diagnose the anomaly types of each node through single-node metrics, which avoids high-dimensional input when all metrics of the database are processed at the same time. 2) In most scenarios, the proposed correlation-aware root cause ranking technology can improve the diagnostic effect. In single-node anomaly, multi-node anomaly, and compound anomaly scenarios, the diagnostic effect of DistDiagnosis can surpass that of DistDiagnosis-w/o-CR in all aspects.

6.4 Analysis of the impact of DistDiagnosis overhead on database performance

To verify the influence of the extra overhead caused by DistDiagnosis on OceanBase performance, we compare transaction processing performance of the database before and after the use of DistDiagnosis. Figure 7 shows the transactions per minute (TPM) of OceanBase during its execution of TPC-C under different conditions. In the figure, OceanBase represents the performance of the database without the use of DistDiagnosis. OceanBase-w-Monitor represents the performance of the database when DistDiagnosis is used to collect performance metrics and no diagnostic operation is performed. It simulates an application scenario in which DistDiagnosis is used but diagnosis is not triggered. As for OceanBase-w-Diagnosis, complete DistDiagnosis diagnosis is performed when the performance metrics are collected each time. The diagnosis operation is undertaken by a node in the cluster, and this simulates a scenario in which a heavy diagnosis task is performed.

The performance overhead of DistDiagnosis to collect performance metrics is minimal across various TPC-C warehouse numbers. Compared with OceanBase, OceanBase-w-Monitor faces performance decline by 0.72% at most, which has almost no influence on performance. Moreover, even if DistDiagnosis performs diagnosis continuously, OceanBase-w-Diagnosis has only a little performance loss. Compared with OceanBase, the performance of OceanBase-w-Diagnosis declines by merely 1.17% at most, which indicates DistDiagnosis does not generate too much extra performance overhead, and it can assist in diagnosis of online systems.

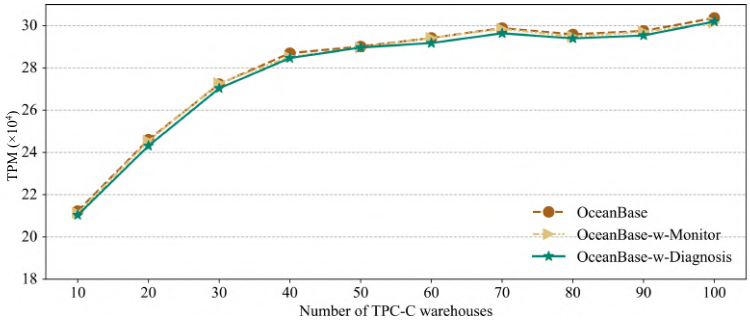

Figure 7 Influence of DistDiagnosis on database performance under different warehouse numbers

6.5 Influence of adding new nodes in cluster on diagnostic effect

To demonstrate the advantage of DistDiagnosis in adapting to the change in database node number, we investigate the diagnostic performance of DistDiagnosis when new nodes are added to the database cluster. The configuration of the newly added nodes is the same as that of the existing nodes in the cluster. Table 8 shows the diagnostic results after new nodes are added. DistDiagnosis-3 stands for the situation that training is conducted using anomaly data from the original three-node cluster, and the diagnosis test is perform on the new cluster. As for DistDiagnosis-4, the node anomaly diagnoser is trained with the anomaly data in the new cluster environment. Other comparison methods cannot be directly transferred and must be trained in the new cluster. The anomaly data in the new cluster are collected in the same way as in Section 6.1 (4).

Table 8 Demonstration of DistDiagnosis after adding a new node

Anomaly scenario	A1			B1			C1			W1		
	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$
RandomForest	0.75	0.75	0.75	0.78	0.78	1.00	0.62	0.62	0.65	0.70	0.98	1.00
XGBoost	0.75	0.75	0.75	0.80	1.00	0.80	0.69	0.69	0.69	0.71	1.00	1.00
DistDiagnosis-3	0.96	1.00	1.00	1.00	1.00	1.00	0.48	0.88	0.90	0.97	1.00	1.00
DistDiagnosis-4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.83**	**0.88**	**0.88**	**1.00**	**1.00**	**1.00**

The results in the table indicate that DistDiagnosis can still maintain a good diagnostic performance when the number of nodes in the cluster changes. In scenarios A1, B1, and W1, the performance of DistDiagnosis-3 which directly migrated from the three-node cluster outperforms other methods. Although its performance in the C1 scenario is slightly inferior, the diagnosis methods designed for the global node metrics need to collect the anomaly data of the new cluster again and conduct retraining when the cluster scale changes. Benefiting from the adopted node-level diagnoser, DistDiagnosis can directly diagnose the newly added nodes, enhancing the generalizability of the method in different anomaly scenarios of the distributed database. Additionally, after the anomaly data in the new cluster are obtained, DistDiagnosis can be retrained to improve accuracy, with DistDiagnosis-4 achieving the best performance in all four scenarios.

6.6 Comparative experiment of diagnostic effect in clusters of different scales

This section shows the diagnostic performance of DistDiagnosis in database clusters of different scales. The node configuration in the clusters is consistent with that in the above experiments, with the node number set as 3, 5, 7, and 9. In the experiment, the adopted four anomaly scenarios are the same as those in Section 6.5. For each cluster size, each diagnostic

method is trained and tested using anomaly data from the corresponding cluster. Figure 8 shows the changes in $AC@k$ of different methods.

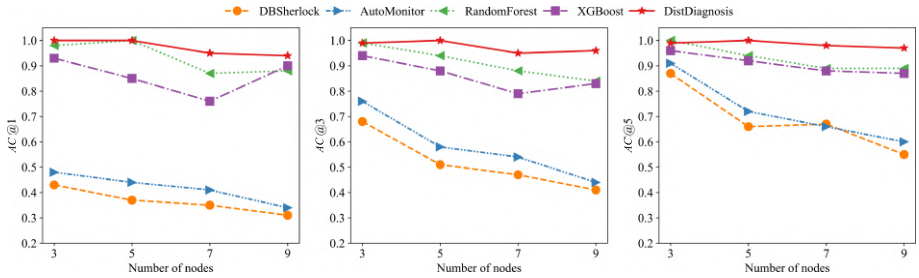

Figure 8 Comparison of diagnostic methods in clusters of different scales

AutoMonitor and DBSherlock perform the worst among all the methods, with their diagnostic performance significantly decreasing as the number of nodes increases. This is because with the increase in the number of corresponding metrics, the classification technologies based on the nearest neighbor or space division are easily influenced by the high dimension of the input variables, which leads to worse classification results. In contrast, DistDiagnosis can still maintain the best diagnostic effect when the number of nodes in the cluster increases gradually. Even in a cluster with 9 nodes, the average $AC@1$, $AC@3$, and $AC@5$ of DistDiagnosis can reach 0.94, 0.96, and 0.97, which are 0.06, 0.12, and 0.08 higher than those of the second-best method RandomForest, respectively. This demonstrates the strong adaptability of the DistDiagnosis method across database clusters of different scales.

6.7 Case analysis

Figure 9 shows examples of the output results of different methods in the case of the multi-node compound anomalies of CPU bottleneck and heavy workload in the database. In the left part of the figure, the weights on the adjacent edges of each node are the edge weights of the compound anomaly graph generated by DistDiagnosis.

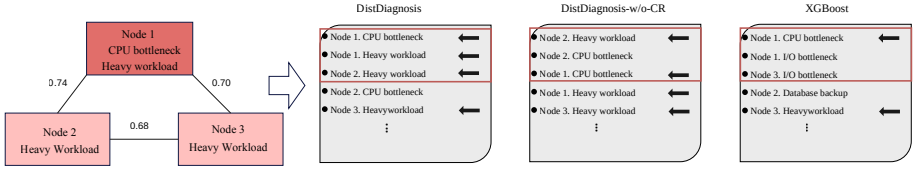

Figure 9 Output results of different methods in the case of multi-node compound anomalies of CPU bottleneck and heavy workload

In this anomaly scenario, all nodes in the database generate heavy workload at the same time. Node 1 additionally generates CPU bottleneck, which is an important node in this anomaly. As shown in the figure, when DistDiagnosis calculates correlations between nodes as the edge weights, the weight of the adjacent edge of Node 1 is the highest, which corresponds to its importance in the current anomaly. Using the correlation-aware root cause ranking technology, DistDiagnosis can adjust the ranking results depending on the importance score of the node when making the final ranking, so as to realize that the top three in the result sequence are all correct root causes. As DistDiagnosis-w/o-CR treats Node 1 and Node 2 equally, it places the wrong anomaly $\langle Node\ 2, CPU\ bottleneck \rangle$ at a higher position. In this case, XGBoost wrongly identifies the I/O bottlenecks on Node 1 and Node 3. DistDiagnosis yields the best results.

7 Summary and Future Work

In this paper, we studied the anomaly diagnosis of distributed databases and proposed a diagnosis method, DistDiagnosis, which can diagnose compound anomalies on nodes of distributed databases. This method uses the compound anomaly graph to model the anomalous state of distributed databases. It can effectively capture the interactions between nodes while representing anomalies on each node. DistDiagnosis introduces a correlation-aware root cause ranking method, which is capable of effectively locating root causes according to the influence of nodes on the whole database. To verify the effectiveness of this method, we constructed anomaly test cases in different scenarios on OceanBase, a domestic distributed database. The experimental results showed that DistDiagnosis is superior to other compared methods, and the highest $AC@1$, $AC@3$, and $AC@5$ reached 0.97, 0.98, and 0.98, which are increased by 5.20%, 5.45%, and 4.46%, respectively, compared to those of the second-best method.

At present, there is still room for DistDiagnosis improvement. First, DistDiagnosis achieves optimal diagnostic performance when applied to nodes with identical hardware specifications. When the resource specifications of nodes differ, DistDiagnosis needs to train different node-level diagnosers for each type of node to handle the variation. The future research will focus on designing a more general diagnosis approach for distributed databases, so that DistDiagnosis can better support the diagnosis of heterogeneous nodes and nodes of different specifications. Additionally, the performance metrics used in this method depend on the selection of database experts by manual experience, which has limitations in evaluating the importance of the metrics. The next step for DistDiagnosis is to design an automatic performance metric selection technique and dynamically adjust the impact of these metrics on the diagnostic results based on their importance.

References

[1] Huang SY, Qin YZ, Zhang XY, Tu YF, Li ZL, Cui B. Survey on performance optimization for database systems. Science China Information Sciences, 2023, 66(2): 121102.

[2] Yoon DY, Niu N, Mozafari B. DBSherlock: A performance diagnostic tool for transactI/Onal databases. Proc. of the 2016 Int'l Conf. on Management of Data. San Francisco: ACM, 2016. 1599–1614. [doi: 10.1145/2882903.2915218]

[3] Jin LY, Li GL. AI-based database performance diagnosis. Ruan Jian Xue Bao/Journal of Software, 2021, 32(3): 845–858 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6177.htm

[4] Wang J, Yang YQ, Wang T, Sherratt RS, Zhang JY. Big data service architecture: A survey. Journal of Internet Technology, 2020, 21(2): 393–405.

[5] Zhang GY, Li CH, Zhou K, Liu L, Zhang C, Chen WC, Fang HT, Cheng B, Yang J, Xing JS. DBCatcher: A cloud database online anomaly detection system based on metric correlation. Proc. of the 2023 IEEE Int'l Conf. on Data Engineering (ICDE). Anaheim: IEEE, 2023. 1126–1139. [doi: 10.1109/ICDE55515.2023.00091]

[6] Huang SY, Wang ZW, Zhang XY, Tu YF, Li ZL, Cui B. DBPA: A benchmark for transactional database performance anomalies. Proc. of the ACM on Management of Data, 2023, 1(1): 72.

[7] Ma MH, Yin Z, Zhang SL, Wang S, Zheng C, Jiang XH, Hu HW, Luo C, Li YL, Qiu NJ, Li FF, Chen CC, Pei D. Diagnosing root causes of intermittent slow queries in cloud databases. Proc. of the VLDB Endowment, 2020, 13(8): 1176–1189.

[8] Cao W, Gao YS, Lin BC, Feng XJ, Xie Y, Lou X, Wang P. TcpRT: Instrument and diagnostic analysis system for service quality of cloud databases at massive scale in real-time. Proc. of the 2018 Int'l Conf. on Management of Data. Houston: ACM, 2018. 615–627. [doi: 10.1145/3183713.3190659]

[9] Pettitt AN. A non-parametric approach to the change-point problem. Journal of the Royal Statistical Society: Series C (Applied Statistics), 1979, 28(2): 126–135.

[10] Dundjerski D, Tomašević M. Automatic database troubleshooting of Azure SQL Databases. IEEE Trans. on Cloud Computing, 2022, 10(3): 1604–1619.

[11] Malhotra P, Ramakrishnan A, Anand G, Vig L, Agarwal P, Shroff G. LSTM-based encoder-decoder for multi-sensor anomaly detessction. arXiv:1607.00148, 2016.

[12] Chen TQ, Guestrin C. XGBoost: A scalable tree boosting system. Proc. of the 22nd Int'l Conf. on Knowledge Discovery and Data Mining. San Francisco: ACM, 2016. 785–794. [doi: 10.1145/2939672.2939785]

[13] Breiman L. Random forests. Machine Learning, 2001, 45(1): 5–32.

[14] Zhou XH, Li GL, Sun ZY, Liu ZY, Chen WZ, Wu JM, Liu JS, Feng RH, Zeng GY. D-bot: Database diagnosis system using large language models. Proc. of the VLDB Endowment, 2024, 17(10): 2514–2527.

[15] Ma M, Xu JM, Wang Y, Chen PF, Zhang ZH, Wang P. AutoMAP: Diagnose your microservice-based Web applications automatically. Proc. of the Web Conf. 2020. Taipei: ACM, 2020. 246–258. [doi: 10.1145/3366423.3380111]

[16] Lu RM, Xu EC, Zhang YM, Zhu FY, Zhu ZS, Wang MT, Zhu ZP, Xue GT, Shu JW, Li ML, Wu JS. Perseus: A fail-slow detection framework for cloud storage systems. Proc. of the 21st USENIX Conf. on File and Storage Technologies. Santa Clara: USENIX Association, 2023. 49–63.

[17] Yang ZK, Yang CH, Han FS, Zhuang MQ, Yang B, Yang ZF, Cheng XJ, Zhao YZ, Shi WH, Xi HF, Yu H, Liu B, Pan Y, Yin BX, Chen JQ, Xu QQ. OceanBase: A 707 million tpmC distributed relational database system. Proc. of the VLDB Endowment, 2022, 15(12): 3385–3397.

[18] Corbett JC, Dean J, Epstein M, et al. Spanner: Google's globally distributed database. ACM Trans. on Computer Systems (TOCS), 2013, 31(3): 8.

[19] Huang DX, Liu Q, Cui Q, Fang ZH, Ma XY, Xu F, Shen L, Tang L, Zhou YX, Huang ML, Wei W, Liu C, Zhang J, Li JJ, Wu XL, Song LY, Sun RX, Yu SP, Zhao L, Cameron N, Pei LQ, Tang X. TiDB: A Raft-based HTAP database. Proc. of the VLDB Endowment, 2020, 13(12): 3072–3084.

[20] Xing W, Ghorbani A. Weighted PageRank algorithm. Proc. of the 2nd Annual Conf. on Communication Networks and Services Research. Fredericton: IEEE, 2004. 305–314. [doi: 10.1109/DNSR.2004.1344743]

[21] Liu P, Zhang SL, Sun YQ, Meng Y, Yang JH, Pei D. FluxInfer: Automatic diagnosis of performance anomaly for online database system. Proc. of the 39th IEEE Int'l Performance Computing and Communications Conf. (IPCCC). Austin: IEEE, 2020. 1–8.

[22] Meng Y, Zhang SL, Sun YQ, Zhang RR, Hu ZL, Zhang YY, Jia CY, Wang ZG, Pei D. Localizing failure root causes in a microservice through causality inference. Proc. of the 28th IEEE/ACM Int'l Symp. on Quality of Service (IWQoS). Hangzhou: IEEE, 2020. 1–10. [doi: 10.1109/IWQoS49365.2020.9213058]

[23] Kim M, Sumbaly R, Shah S. Root cause detection in a service-oriented architecture. ACM SIGMETRICS Performance Evaluation Review, 2013, 41(1): 93–104.

[24] Benesty J, Chen JD, Huang YT, Cohen I. Pearson correlation coefficient. In: Noise Reduction in Speech Processing. Berlin: Springer, 2009. 1–4. [doi: 10.1007/978-3-642-00296-0_5]

[25] Xu Y, Kostamaa P, Zhou X, Chen L. Handling data skew in parallel joins in shared-nothing systems. Proc. of the 2008 ACM SIGMOD Int'l Conf. on Management of Data. Vancouver: ACM, 2008. 1043–1052. [doi: 10.1145/1376616.1376720]

[26] Zilio DC, Jhingran A, Padmanabhan S. Partitioning key selection for a shared-nothing parallel database system. 1994. https://api.semanticscholar.org/CorpusID:14485315

[27] Kumar TVV, Kumar A, Singh R. Distributed query plan generation using particle swarm optimization. Int'l Journal of Swarm Intelligence Research (IJSIR), 2013, 4(3): 58–82.

[28] Wang P, Xu JM, Ma M, Lin WL, Pan DS, Wang Y, Chen P. CloudRanger: Root cause identification for cloud native systems. Proc. of the 18th IEEE/ACM Int'l Symp. on Cluster, Cloud and Grid Computing (CCGRID). Washington: IEEE, 2018. 492–502.

Qingfeng Xiang, master's degree candidate. His research interests include cross-technologies of distributed systems, database systems, and machine learning.

Quanqing Xu, Ph.D., senior engineer. His research interests include database systems and distributed systems.

Yingxia Shao, Ph.D., professor, doctoral supervisor. His research interests include efficient computation and learning of large-scale graph data, parallel computing frameworks, and knowledge graph analysis.

Chuanhui Yang, master. His research interests include distributed systems and database systems.