TaurusDB for PostgreSQL

Troubleshooting

Issue	01
Date	2025-11-14

HUAWEI

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: https://www.huaweicloud.com/intl/en-us/

Contents

1 A Large Number of Schemas Whose Owner Is rdsadmin

Scenario

There are a large number of schemas whose owner is **rdsadmin** in a TaurusDB for PostgreSQL instance.

Possible Causes

Temporary tables in a TaurusDB for PostgreSQL instance are classified into session-level temporary tables and transaction-level temporary tables.

- In a session-level temporary table, data exists throughout the lifecycle of a session. The default temporary table is on a per-session basis.
- In a transaction-level temporary table, data exists only in the lifecycle of a transaction.

TaurusDB for PostgreSQL temporary tables are special tables generated in a schema named **pg_temp**_*n*, where *n* indicates a number that varies by session.

In this scenario, a large number of temporary tables are used by user services. These temporary tables cannot be deleted. If they are deleted, they will be re-created soon.

2 Authentication Not Supported When a DB Instance Is Accessed Through an Application

Scenario 1

- Description

  If a TaurusDB for PostgreSQL instance is connected through a PostgreSQL application and the client does not support the scram-sha-256 authentication method, the following error message is displayed:

  ```
  Authentication method not supported (Received: 10)
  ```

- Possible causes

  The client version is too old and incompatible with the encryption algorithm used by the DB instance.

- Solution

  a. Check the client or client driver (such as the JDBC driver) and update it to the latest version to ensure that the latest authentication method is supported.

  If the latest authentication is still not supported, go to the next step.

  b. Change the value of **password_encryption** to **md5**.

  NOTICE

  The modification of **password_encryption** takes effect only after the password is reset.

  c. If the fault persists, check whether the authentication method is set to **scram-sha-256**. If yes, change it to **md5** and try again.

Scenario 2

- Description

  After data is migrated from an instance of an earlier version to an instance of a later version, the new instance cannot be connected and the following error message is displayed:


```
unsupported authentication method requested by the server: 10
```

- Possible causes

  The value of **password_encryption** is **md5** on the original instance. After data is migrated to the new instance, the default value of **password_encryption** becomes **scram-sha-256**. This value is changed to **md5**, but the password is not reset after the change, so the password authentication fails.

- Solution

  Reset the password after the value of **password_encryption** is changed.

Scenario 3

- Description

  When a TaurusDB for PostgreSQL instance is connected using a JDBC application, the following error message is displayed in the console error logs because the JDBC version does not support the scram-sha-256 authentication:

```
unsupported frontend protocol 1234.5680: server supports 2.0 to 3.0
```

- Possible causes

  The JDBC version is too early.

- Solution

  Download and use a JDBC of the latest version.

3 Error Reported When a Request Is Executed Through an Existing Connection

Scenario

When a client executes a request through an existing connection, the following error information is displayed:

- Error 1
  ```
  org.postgresql.util.PSQLException: An I/O error occurred while sending to the backend.
  ```
- Error 2
  ```
  org.postgresql.util.PSQLException: The connection attempt failed
  ...
  Caused by: java.net.SocketException: Connection reset
  ```

Solution to Error 1

Possible causes:

- There are too many parameters in the SQL statement.
- The connection has been released.

Solution:

- If there are too many parameters in the SQL statement, split the SQL statement.
- If the connection has been released, check the client connection parameters, such as the connection timeout interval. Add an automatic retry mechanism for your client.

Solution to Error 2

This error is reported when a connection that has been released is used. The possible causes are as follows:

- There are problems with network connectivity.
- The instance is being rebooted or the backend process crashes.
- Idle connections are released upon timeout.

Solution:

1. Check the network connectivity and determine whether the disconnection is caused by network link factors (such as high packet loss rate and retransmission rate).
2. If there are no network problems, check for other errors.
3. If no, check the connection timeout parameters (such as **sockettimeout** and **connecttimeout** for the JDBC connection pool). If the values are too small, connections can be released automatically.

4 Slow Instance Reboot Due to Too Many Inodes

Excessive inodes are often caused by stacked temporary files or excessive table objects. If there are too many inodes, the instance can reboot slowly.

Scenario 1

- Description

  When a large number of complex SQL statements are executed on a TaurusDB for PostgreSQL instance, temporary files are stacked and an out of memory (OOM) exception occurs. In this case, the instance reboots slowly and services are unavailable for a long time.

- Possible causes

  If the SQL statements involve operations such as sorting, hash joins, and aggregation and the memory usage exceeds the value of **work_mem**, temporary files are generated. If a large number of such SQL statements are executed, an OOM exception occurs and the OS kills the database process. The kernel does not clear the temporary files, causing stacked temporary files. Too many temporary files slow down the database startup. That is because all temporary files need to be deleted before the database process can start.

- Solution

  Optimize the SQL statements or increase the value of **work_mem** to reduce the number of temporary files generated.

Scenario 2

- Description

  There are a large number of tables in a TaurusDB for PostgreSQL instance. At a certain time, connections to the instance and workloads increase sharply. The database process memory is used up and an OOM issue occurs. Then the instance reboots slowly, and services are unavailable for a long time.

- Possible causes

  During the reboot, the kernel process traverses all tables and flushes data in the OS cache to disks (fsync). If there are too many table objects, the traversal takes a long time and slows down the reboot.

- Solution
  - Ensure that there are no more than 20,000 tables in a single instance and no more than 4,000 tables in a single database.
  - Configure memory monitoring on the application side to prevent OOM issues. Pay attention to the value of metric **Inodes** and control the number of objects.

