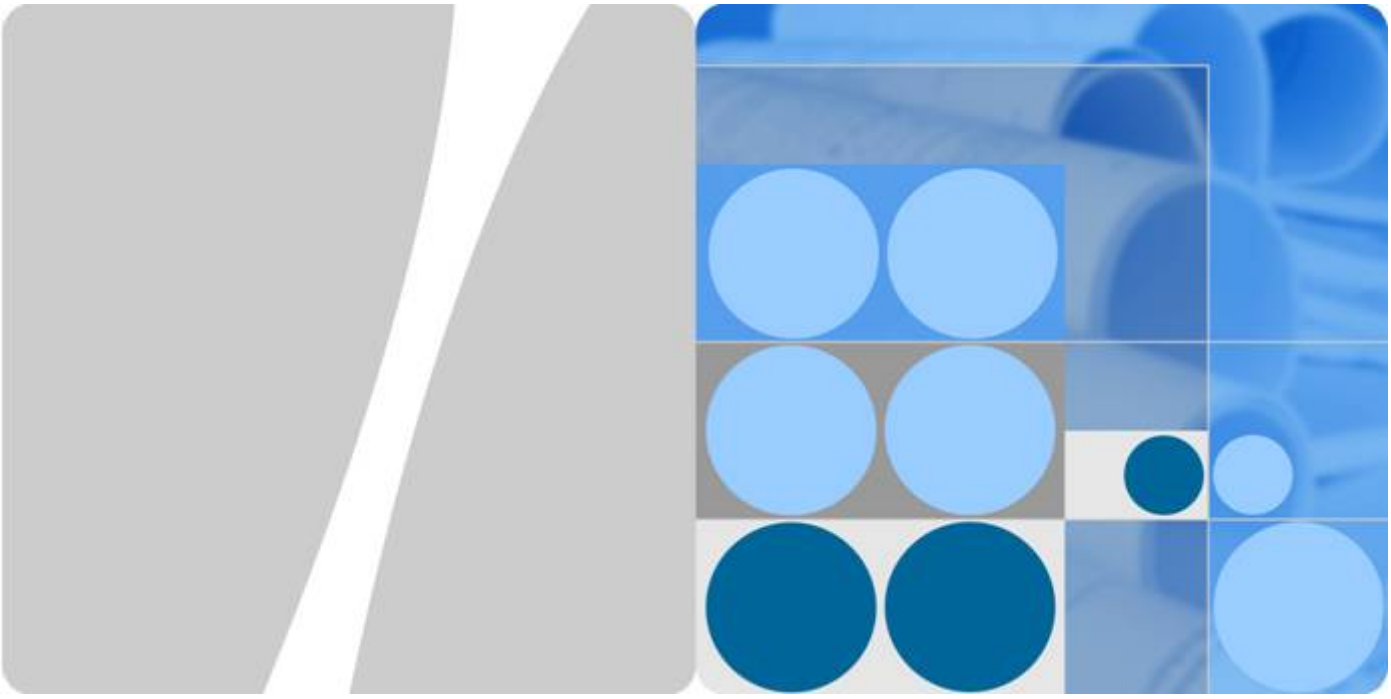


Relational Database Service

HUAWEI CLOUD MySQL Kernel

Issue 03

Date 2021-01-21

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

Copyright © Huawei Technologies Co., Ltd. 2021. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

Trademarks and Permissions


and other Huawei trademarks are trademarks 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 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 Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: https://e.huawei.com

Contents

1 Kernel Version Description....1
2 Connection Thread Pool....8
3 Dual Channels for Primary/Standby Replication....12
4 MDL Views....14
5 Online Varchar Length Increase....18
6 Ending Idle Transactions....19
A Change History....23

 Copyright © Huawei Technologies Co., Ltd.

1 Kernel Version Description

This section describes the kernel version updates of RDS for MySQL.

MySQL 8.0

Table 1-1 RDS for MySQL 8.0 version description

Date	Description
2020-12-31	• Performance optimized The compiler was upgraded to GCC 9. • Issues resolved MySQL start failures in hugepage memory were resolved.
2020-12-01	• Performance optimized The efficiency of collecting additional information about slow query logs was improved. • Issues resolved The issue that replication on the standby DB instance may be interrupted during the rollback of distributed transactions was resolved.
2020-11-06	• New features Hugepage memory was supported. • Issues resolved Timing error caused by gettimeofday in multiple threads was fixed.
2020-09-21	• Disconnection details were recorded in error logs. • Index hints were supported.
2020-08-03	• The execution time and waiting time of large transactions were displayed. • Independent connection control was used to manage users. • SQL filter was used to restrict the execution frequency of SQL statements during peak hours. • Kernel performance was optimized.

 Copyright © Huawei Technologies Co., Ltd.

Date	Description
2020-06-19	• The kernel version was upgraded to 8.0.20. • Kernel performance was optimized.
2020-02-15	• MySQL 8.0 was put into commercial use. • The HUAWEI CLOUD Kunpeng-powered Arm kernel version was released.
2019-12-15	• The kernel version was upgraded to 8.0.17. • The parallel index creation is 2.5 times faster.
2019-10-15	The issue that the replication delay is abnormal when **SQL_MODE** is set to **PAD_CHAR_TO_FULL_LENGTH** was resolved.
2019-09-15	• Thread pools were supported. • OpenSSL was upgraded to 1.1.1a. • The Common Type System (CTS) syntax **create table xx select** was supported. • The memory usage and CPU time usage of user threads can be queried through **show full processlist**.

MySQL 5.7

Table 1-2 RDS for MySQL 5.7 version description

Date	Description
2020-12-31	• Performance optimized – The efficiency of collecting additional information about slow query logs was improved. – The compiler was upgraded to GCC 9. • Issues resolved The issue that replication on the standby DB instance may be interrupted during the rollback of distributed transactions was resolved.
2020-12-01	The conflict frequency between fil_sys mutual exclusions was reduced.
2020-11-06	• New features – Hugepage memory was supported. – Transparent data encryption was supported. – The compiler was optimized. – The UTF-8 encoding efficiency was improved on non-Arm platforms. • Issues resolved Timing error caused by gettimeofday in multiple threads was fixed.

 Copyright © Huawei Technologies Co., Ltd.

Date	Description
2020-09-21	• The kernel version was upgraded to 5.7.31. • SQL filter was used to restrict the execution frequency of SQL statements during peak hours.
2020-08-03	• Kernel performance was optimized. • The recycle bin was supported. • Issues that may occur during local disk data cleanup were resolved.
2020-07-09	• Kernel performance was optimized. • Users' operation history can be recorded in error logs. • Distributed transaction stability was improved.
2020-06-30	• Kernel performance was optimized. • Local disk logs whose size exceeds the local disk capacity were stored on cloud disks. • The buffer pool memory initialization module was optimized to improve the initialization efficiency. • The thread security of some operations on Arm was improved.
2020-05-30	• New features – Index hints were supported. – Full SQL logs can be captured. • Issues resolved The issue of occasional database connection failures was resolved.
2020-04-30	Kernel performance was optimized.
2020-03-30	• The kernel version was upgraded to 5.7.29. • Kernel performance was optimized. • Threshold pools supported statistics on I/O waits.
2020-02-15	• Replication dual-channel: A replication status channel was added to accurately determine the replication status when the primary database crashes, ensuring that transactions are not lost. • Optimized ROW_IMAGE mode: The binlog size was reduced, and migration and SQL flashback were supported.

 Copyright © Huawei Technologies Co., Ltd.

Date	Description
2019-12-15	• DDL operation progress: The progress of adding columns or indexes was obtained through **information_schema.innodb_alter_table_progress**. • Long transactions: The transaction execution time **Trx_Executed_Time** can be obtained through **show processlist**. • Online extension of the **string** field length: The **VARCHAR** field length was changed from **COPY** to **INPLACE** by default. • Abundant InnoDB deadlock information: The complete onsite information when deadlock occurs at the InnoDB layer can be viewed through **show engine innodb status**.
2019-10-15	• Performance optimized – The HUAWEI CLOUD Kunpeng-powered Arm kernel version was released. • New features – The kernel version was upgraded to 5.7.27. – Instant add column: Columns are quickly added to a table without copying data and without occupying disk space and disk I/Os. Data can be updated in real time during peak hours. – Metadata lock (MDL) view: The MDL information held or waited by the thread can be obtained through **information_schema.metadata_lock_info**.
2019-08-15	Jemalloc memory management: The jlibc memory management module was replaced to reduce memory usage and improve memory allocation efficiency.
2019-06-15	• The kernel version was upgraded to 5.7.25. • Implicit primary key: If a user does not specify a primary key or unique index when creating a table, an implicit primary key is automatically created. The primary/standby replication delay caused by no primary key is resolved.

 Copyright © Huawei Technologies Co., Ltd.

Date	Description
2019-05-15	• New features – When **sync_binlog** and **innodb_flush_log_at_trx_commit** are set to values other than **1**, the crash safe data security of the standby database is guaranteed. In sysbench high-concurrent write-only request scenarios, the replication delay between primary and standby DB instances is almost zero. • Issues resolved – The issue that when **relay_log_recovery** is set to **ON**, the standby database cannot be rebooted after being killed in certain scenarios was resolved. – The issue that the replication delay is abnormal when **SQL_MODE** is set to **PAD_CHAR_TO_FULL_LENGTH** was resolved. – The issue that **performance_schema** statistics are repeatedly measured was resolved. – The issue that when **ORDER BY** queries are performed on tables about information replication in **Performance_schema**, the return value is empty was resolved.
2019-01-15	• The issue that data is inconsistent and data replication is interrupted after the flush operation is performed on read replicas was resolved. • The issue that the replication thread of the standby database is suspended due to statements like RREPAIR or OPTIMIZE was resolved.
2018-11-15	• The kernel version was upgraded to 5.7.23. • Temporary tables can be created or deleted in transactions when GTID is enabled. • Table-level multi-threaded slaves (MTS) parallel playback was supported.
2018-07-15	• The kernel version was upgraded to 5.7.22. • Thread pools were supported. • The Common Type System (CTS) syntax **create table xx select** was supported. • Operator pushdown: The aggregation operator is pushed down to the storage engine layer to improve the execution speed of count() and sum(). • Kill idle transactions: Transactions that have been idled for a long time are automatically killed after the timeout threshold is reached. • The memory usage and CPU time usage of user threads can be queried through **show full processlist**.

 Copyright © Huawei Technologies Co., Ltd.

MySQL 5.6

Table 1-3 RDS for MySQL 5.6 version description

Date	Description
2020-12-31	SQL filter was reconstructed to improve usability.
2020-11-06	The kernel version was upgraded to 5.6.50.
2020-09-23	SQL filter was used to restrict the execution frequency of SQL statements during peak hours.
2020-08-03	The kernel version was upgraded to 5.6.49.
2020-07-09	• Local disk logs whose size exceeds the local disk capacity were stored on cloud disks. • Users' operation history can be recorded in error logs.
2020-05-30	The buffer pool memory initialization module was optimized to improve the initialization efficiency.
2020-04-30	The issue of occasional database connection failures was resolved.
2020-03-30	• Full SQL collection was supported. • The compiler was upgraded to 7.3. • The kernel version was upgraded to 5.6.47.
2020-02-15	• Replication dual-channel: A replication status channel was added to accurately determine the replication status when the primary database crashes, ensuring that transactions are not lost. • Optimized ROW_IMAGE mode: The binlog size was reduced, and migration and SQL flashback were supported.
2019-12-15	• Long transactions: The transaction execution time **Trx_Executed_Time** can be obtained through **show processlist**. • Online extension of the **string** field length: The **VARCHAR** field length was changed from **COPY** to **INPLACE** by default. • Abundant InnoDB deadlock information: The complete onsite information when deadlock occurs at the InnoDB layer can be viewed through **show engine innodb status**.
2019-10-15	• The kernel version was upgraded to 5.6.45. • The memory usage and CPU time usage of user threads can be queried through **show full processlist**. • Kill idle transactions: Transactions that have been idled for a long time are automatically killed after the timeout threshold is reached.

Copyright © Huawei Technologies Co., Ltd.

Date	Description
2019-08-15	Jemalloc memory management: The jlibc memory management module was replaced to reduce memory usage and improve memory allocation efficiency.
2019-06-15	• The kernel version was upgraded to 5.6.43. • The audit function was supported. • Implicit primary key: If a user does not specify a primary key or unique index when creating a table, an implicit primary key is automatically created to resolve the primary/standby replication delay issue.
2019-05-15	• The issue that the replication delay **Second_Behind_Master** is inaccurate in certain scenarios was resolved. • The issue that the replication delay is abnormal when **SQL_MODE** is set to **PAD_CHAR_TO_FULL_LENGTH** was resolved.
2019-01-15	• The issue that data is inconsistent and data replication is interrupted after the flush operation is performed on read replicas was resolved. • The issue that the replication thread of the standby database is suspended due to statements like RREPAIR or OPTIMIZE was resolved. • The issue that an error occurs on the **grant select(column_name)** statement replication threshold was resolved.
2018-11-15	• The kernel version was upgraded to 5.6.41. • Temporary tables can be created or deleted in transactions when GTID is enabled. • Table-level multi-threaded slaves (MTS) parallel playback was supported.
2018-07-15	• New features – The kernel version was upgraded to 5.6.40. – Thread pools were supported. – The Common Type System (CTS) syntax **create table xx select** was supported. • Issues resolved – The issue that binlog and relay log names depend on the PID file names was resolved. – The issue that the **relay_log_basename** parameter is not specified, resulting in the failure of primary/standby replication was resolved. – The issue that the **force index** syntax becomes invalid in the **group_by xx order_by xx limit n1,n2** scenario was resolved.

 Copyright © Huawei Technologies Co., Ltd.

2 Connection Thread Pool

Introduction

When there are a large number of concurrent database connections, a large number of resources are occupied, and the performance of the MySQL server deteriorates significantly. RDS for MySQL provides a connection thread pool that uses a few active threads to serve a large number of database connections. This decouples connections from execution and improves database performance in high-concurrency scenarios.

Characteristics

RDS for MySQL connection thread pool provides the following benefits:

- A large number of database connections can be processed, and resource contention and context switches are reduced.
- The number of concurrent transactions is limited. When the database load is heavy, transactions that are being executed are preferentially guaranteed.
- Connections are processed quickly to prevent thread exceptions.
- When a transaction is waiting for I/Os and locks, CPU resources and other connections are released.

Thread Pool Operations

- Querying thread pool parameters

  Run **show variables** to query thread pool parameters.

```
show variables like 'threadpool%';
+-----------------------------------+------------+
| Variable_name                     | Value      |
+-----------------------------------+------------+
| threadpool_enabled                | ON         |
| threadpool_high_prio_tickets      | 4294967295 |
| threadpool_idle_timeout           | 60         |
| threadpool_long_conn_time         | 2          |
| threadpool_max_threads            | 100000     |
| threadpool_oversubscribe          | 3          |
| threadpool_prio_kickup_timer      | 1000       |
| threadpool_rec_launch_time        | ON         |
| threadpool_size                   | 128        |
| threadpool_slow_conn_log          | ON         |
| threadpool_slow_conn_log_interval | 30         |
```

```
| threadpool_slow_launch_time       | 2          |
| threadpool_stall_limit            | 500        |
+-----------------------------------+------------+
```

Table 2-1 Thread pool parameters

Parameter	Description
threadpool_enabled	Enables or disables thread pools.
threadpool_high_prio_tickets	Number of tickets held by a high-priority thread.
threadpool_idle_timeout	Idle time before a thread is destroyed, in seconds.
threadpool_long_conn_time	If the login time exceeds the value of this parameter, the login information is printed in logs.
threadpool_max_threads	Maximum number of threads that can be created in a thread pool.
threadpool_oversubscribe	Maximum number of extra threads that can be created in a thread group.
threadpool_prio_kickup_timer	Maximum duration (in milliseconds) in a low-priority queue.
threadpool_rec_launch_time	Records the thread launch time.
threadpool_size	Number of thread groups.
threadpool_slow_conn_log	Whether to record slow logins in error logs.
threadpool_slow_conn_log_interval	Recording frequency. After a slow login is recorded, the system does not record logins within this interval.
threadpool_slow_launch_time	If the login or query time is greater than the value of this parameter, the value of **threadpool_slow_launch_request** in status increases by 1.
threadpool_stall_limit	Interval for checking whether a thread group is busy.

 Copyright © Huawei Technologies Co., Ltd.

Table 2-2 Thread pool parameters that can be modified

Parameter	Dynamic Parameter	Data Type	Value Range	Description
threadpool_enabled	Yes	boolean	[ON,OFF]	• **ON**: Enables the thread pool. • **OFF**: Disables the thread pool.
threadpool_oversubscribe	Yes	integer	[1,50]	Maximum number of extra threads that can be created in a thread group.
threadpool_size	Yes	integer	[1,512]	Number of thread groups.

- Querying thread pool status

  Run **show status** to query the thread pool status.

```
show status like 'threadpool%';
+-------------------------------------+------------+
| Variable_name                       | Value      |
+-------------------------------------+------------+
| Threadpool_active_connections       | 65         |
| Threadpool_active_threads           | 5          |
| Threadpool_avg_launch_time          | 0          |
| Threadpool_dump_threads             | 0          |
| Threadpool_idle_threads             | 63         |
| Threadpool_running                  | ON         |
| Threadpool_slow_launch_request      | 0          |
| Threadpool_threads                  | 237        |
| Threadpool_threads_high_water_mark | 1075       |
| Threadpool_waiting_threads          | 57         |
| Threadpool_worst_launch_time        | 692548     |
+-------------------------------------+------------+
```

Table 2-3 Thread pool status

Status	Description
Threadpool_active_connections	Number of active connections in a thread pool
Threadpool_active_threads	Number of active threads in a thread pool
Threadpool_avg_launch_time	Average waiting time, in milliseconds
Threadpool_dump_threads	Number of dump threads

 Copyright © Huawei Technologies Co., Ltd.

Status	Description
Threadpool_idle_thread	Number of idle threads in a thread pool
Threadpool_running	Whether a thread pool is running
Threadpool_slow_launch_request	Number of times that the slow_launch_request is exceeded
Threadpool_threads	Total number of connections in a thread pool
Threadpool_threads_high_water_mark	Number of historical high threads
Threadpool_waiting_threads	Status of the waiting thread pool
Threadpool_worst_launch_time	Worst launch time, in milliseconds

Performance Tests

Table 2-4 Performance tests of different threads

Model	Threads	Thread Pool Enabled	QPS	Latency (ms)
oltp_update_non_index	32	Yes	5932.47	7.84
oltp_update_non_index	64	Yes	10074.11	9.39
oltp_update_non_index	128	Yes	18079.61	10.65
oltp_update_non_index	256	Yes	27439.38	14.46
oltp_update_non_index	512	Yes	33007.96	28.16
oltp_update_non_index	1024	Yes	30282.13	51.94
oltp_update_non_index	2048	Yes	29836.86	95.81

 Copyright © Huawei Technologies Co., Ltd.

3 Dual Channels for Primary/Standby Replication

This section describes the status notification and status replication channel. This function is available only for primary/standby DB instances running MySQL 5.7 or 5.6 and using the semi-synchronous replication mode.

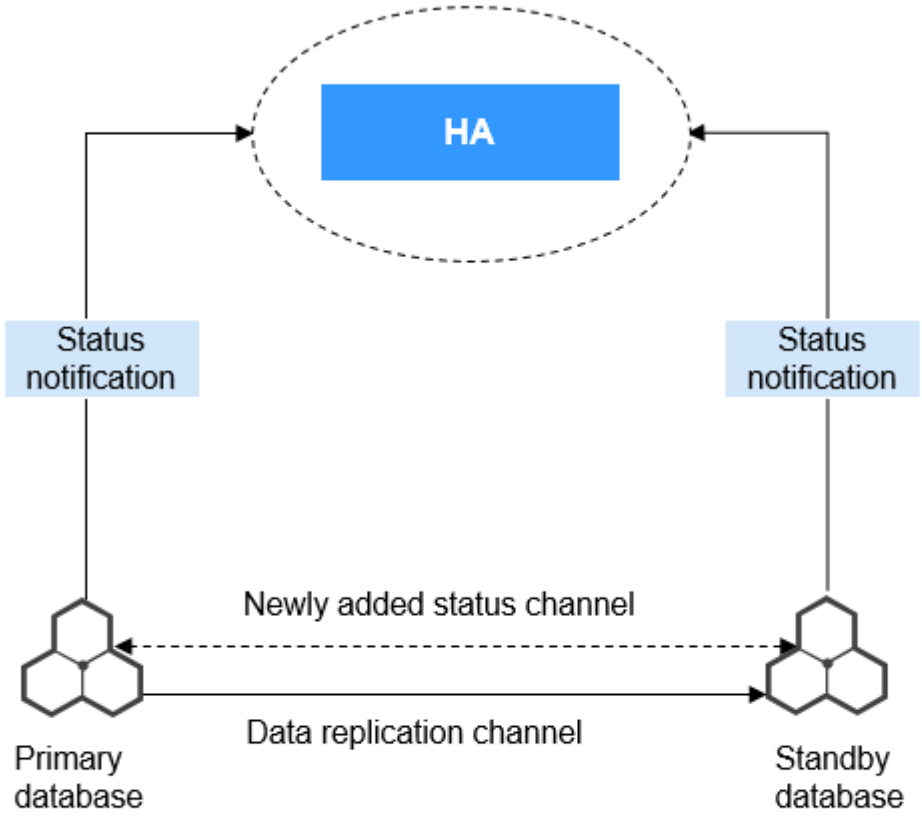
Currently, dual-channel primary/standby replication is available. To enable this function, choose **Service Tickets > Create Service Ticket** in the upper right corner of the management console to submit the application.

Introduction

RDS for MySQL reliable replication is a feature developed by the HUAWEI CLOUD database team. The primary/standby replication is a popular deployment mode of HUAWEI CLOUD MySQL, and semi-synchronous replication is usually used. Based on semi-synchronous replication, the replication status when the primary database crashes can be determined through the status channel and status notification. Both the management nodes (HA) and standby database store the last replication status of the primary database. With dual-channel replication, the efficiency of the primary database is maximized. If the primary database fails, services are quickly switched to the standby database. In addition, data in consistent between the primary and standby databases, ensuring high data reliability.

 Copyright © Huawei Technologies Co., Ltd.

Figure 3-1 Dual-channel architecture



 Copyright © Huawei Technologies Co., Ltd.

4 MDL Views

Introduction

MySQL Community Edition cannot obtain details about table MDLs. If "Waiting for metadata lock" is displayed, blocking DML or DDL, you cannot identify the association among sessions. The DB instance may need to be restarted to solve this problem in some cases, affecting proper service running.

In complex service scenarios, such problems will frequently occur if exclusive operations like DDL and LOCK Table are performed on database metadata, bringing troubles to you.

To resolve the problems, HUAWEI CLOUD MySQL introduces the MDL view, enabling you to view MDLs that each session is holding and waiting for. You can effectively diagnose the system and identify the problematic sessions, minimizing the impact on services.

Description

The MDL view is displayed as a system table. The table is named **metadata_lock_info** and contained in the **information_schema** database. The table structure is as follows.

```
desc information_schema.metadata_lock_info;
+---------------+----------------------+---------+------+----------+--------+
| Field         | Type                 | Null    |Key   |Default   |  Extra |
+---------------+----------------------+---------+------+----------+--------+
| THREAD_ID     | bigint(20) unsigned  | NO      |      |   0      |        |
| LOCK_STATUS   | varchar(24)          | NO      |      |          |        |
| LOCK_MODE     | varchar(24)          | YES     |      |  NULL    |        |
| LOCK_TYPE     | varchar(30)          | YES     |      |  NULL    |        |
| LOCK_DURATION | varchar(30)          | YES     |      |  NULL    |        |
| TABLE_SCHEMA  | varchar(64)          | YES     |      |  NULL    |        |
| TABLE_NAME    | varchar(64)          | YES     |      |  NULL    |        |
+---------------+----------------------+---------+------+----------+--------+
```

 Copyright © Huawei Technologies Co., Ltd.

Table 4-1 metadata_lock_info fields

No.	Field Name	Type	Description
0	THREAD_ID	bigint(20) unsigned	Session ID.
1	LOCK_STATUS	varchar(24)	Two statuses of MDL: ● **PENDING**: The session is waiting for the MDL. ● **GRANTED**: The session has obtained the MDL.
2	LOCK_MODE	varchar(24)	MDL mode, such as MDL_SHARED, MDL_EXCLUSIVE, MDL_SHARED_READ, and MDL_SHARED_WRITE.
3	LOCK_TYPE	varchar(30)	MDL type, such as Table metadata lock, Schema metadata lock, Global read lock, and Tablespace lock.
4	LOCK_DURATION	varchar(30)	MDL range. The value options are as follows: ● **MDL_STATEMENT**: statement-level MDLs ● **MDL_TRANSACTION**: transaction-level MDLs ● **MDL_EXPLICIT**: global-level MDLs
5	TABLE_SCHEMA	varchar(64)	Database name. For some global-level MDLs, this parameter is left empty.
6	TABLE_NAME	varchar(64)	Table name. For some global-level MDLs, this parameter is left empty.

Examples

Scenario: If no transaction is committed for a long time, DDL operations are blocked, and then all operations on the same table are blocked.

Table 4-2 MDL view example

Table Name	Session			
	Session 2	Session 3	Session 4	Session 5
t1	begin; select * from t1;	-	-	-

 Copyright © Huawei Technologies Co., Ltd.

t2	-	begin; select * from t2;	-	-
t3	-	-	truncate table t2; (blocked)	-
t4	-	-	-	begin; select * from t2; (blocked)

Case Analysis

Description

After TRUNCATE operations on table t2 are blocked, SELECT operations on table t2 are also blocked in the service process.

Problem Analysis

- **Without the MDL view**

  If DDL operations are blocked, run the **show processlist** command. Information in the following figure is displayed.

```
show processlist;
+------+--------+--------------+--------+-----------+----------
+-----------------------------------|-------------------------|
| Id   | User   |  Host        |  db    |  Command  |   Time   |   State                        |Info                    |
+---------------+-----------------------+-----------+----------+-----------------------------------
+-------------------------|
| 2    | root   |  localhost   |  test  |  Sleep    |   73     |                                | Null                   |
| 3    | root   |  localhost   |  test  |  Sleep    |   63     |                                | Null                   |
| 4    | root   |  localhost   |  Null  |  Query    |   35     | Waiting for table metadata lock   | truncate
table test.t2  |
| 5    | root   |  localhost   |  test  |  Query    |   17     | Waiting for table metadata lock   | select * from
test.t2   |
| 6    | root   |  localhost   |  test  |  Query    |    0     | starting                          | show processlist       |
+------+--------+--------------+--------+-----------+----------
+-----------------------------------|-------------------------|
```

  According to the preceding thread list:

  – When executing TRUNCATE, session 4 is blocked by the table metadata lock held by other sessions.

  – When executing SELECT, session 5 is also blocked by the table metadata lock held by other sessions.

  – You cannot determine which session blocks session 4 and session 5.

  In this case, killing other sessions randomly will cause great risks to online services. Therefore, you can only wait for other sessions to release the MDL.

- **With the MDL view**

  Run the **select * from information_schema.metadata_lock_info** command to view the MDL information. The following information is displayed.

```
select * from information_schema.metadata_lock_info;
+-------------+-------------+-------------------------+----------------------+-------------------+----------------
```

```
+----------------+
| THREAD_ID   | LOCK_STATUS |  LOCK_MODE              | LOCK_TYPE           |  LOCK_DURATION    |
TABLE_SCHEMA  |   TABLE_NAME   |
+-------------+-------------+-------------------------+---------------------+-------------------+----------------
+----------------+
| 2           | GRANTED     |  MDL_SHARED_READ        | Table metadata lock  |  MDL_TRANSACTION  |
test          |   t1          |
| 3           | GRANTED     |  MDL_SHARED_READ        | Table metadata lock  |  MDL_TRANSACTION  |
test          |   t2          |
| 4           | GRANTED     |  MDL_INTENTION_EXCLUSIVE | Global read lock     |  MDL_STATEMENT
|               |               |
| 4           | GRANTED     |  MDL_INTENTION_EXCLUSIVE | Schema metadata lock |
MDL_TRANSACTION  |  test          |               |
| 4           | GRANTED     |  MDL_EXCLUSIVE          | Table metadata lock  |                  |  test         |
t2          |
| 5           | GRANTED     |  MDL_SHARED_READ        | Table metadata lock  |                  |  test
|   t2          |
+-------------+-------------+-------------------------+---------------------+-------------------+----------------
+----------------+
```

The **show processlist** command output shows information about threads and MDL views.

- Session 4 is waiting for an MDL on table t2.
- Session 3 holds a transaction-level MDL on table t2. If the transaction hold by session 3 is not committed, session 4 will be kept blocked.

You only need to run the **commit** command on session 3 or kill session 3 to keep services running.

 Copyright © Huawei Technologies Co., Ltd.

5 Online Varchar Length Increase

Introduction

Varchar is a set of character data. The native MySQL only supports varchar whose length is no more than 256 bytes. To increase its length to more than 256 bytes, you need to copy data to new tables and lock the tables to prevent data writes during the length increase. HUAWEI CLOUD MySQL has no limitations on the varchar length and allows you to increase it online.

Category

- Increase the varchar length to no more than 256 bytes.

```
create table t1(a varchar(10));
Query OK, 0 rows affected (0.03 sec)
alter table t1 modify a varchar(100),ALGORITHM=INPLACE, LOCK=NONE;
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warning: 0
```

- Increase the varchar length from less than 256 bytes to more than 256 bytes.

```
create table t1(a varchar(100));
Query OK, 0 rows affected (0.05 sec)
alter table t1 modify a varchar(300),ALGORITHM=INPLACE, LOCK=NONE;
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warning: 0
```

- Increase the varchar length beyond 256 bytes.

```
create table t1(a varchar(300));
Query OK, 0 rows affected (0.08 sec)
alter table t1 modify a varchar(500),ALGORITHM=INPLACE, LOCK=NONE;
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warning: 0
```

 Copyright © Huawei Technologies Co., Ltd.

6 Ending Idle Transactions

Introduction

If an idle transaction is not committed for a long time, its rollback will consume database resources and performance. If a large number of idle transactions are not committed and not rolled back for a long time, the performance loss to a database is severe especially during peak hours.

Characteristics

HUAWEI CLOUD MySQL supports idle transaction disconnection when a rollback timed out. This function has the following characteristics:

- Different parameters are used to control different types of transactions.
- When idle transactions timed out, they are automatically rolled back and disconnected.

Description

Run **show variables** to query related parameters.

```
show variables like '%idle%';
+-----------------------------------+-------+
| Variable_name                     | Value |
+-----------------------------------+-------+
| idle_readonly_transaction_timeout | 0     |
| idle_transaction_timeout          | 0     |
| idle_write_transaction_timeout    | 0     |
+-----------------------------------+-------+
```

 Copyright © Huawei Technologies Co., Ltd.

Table 6-1 Parameter description

Parameter	Type	Value Range	Dynamic Validation	Description
idle_readonly_transaction_timeout	integer	Positive integer	Yes	Time in seconds that the server waits for idle read-only transactions before killing the connection. If this parameter is set to **0**, there is not timeout threshold for idle read-only transactions.
idle_transaction_timeout	integer	Positive integer	Yes	Time in seconds that the server waits for common idle transactions before killing the connection. If this parameter is set to **0**, there is not timeout threshold for common idle transactions.
idle_write_transaction_timeout	integer	Positive integer	Yes	Time in seconds that the server waits for idle read/write transactions before killing the connection. If this parameter is set to **0**, there is not timeout threshold for idle read/write transactions.

Application Scenarios

The parameters are set as follows.

```
show variables like '%idle%';
+-----------------------------------+-------+
| Variable_name                     | Value |
+-----------------------------------+-------+
| idle_readonly_transaction_timeout | 5     |
| idle_transaction_timeout          | 10    |
| idle_write_transaction_timeout    | 15    |
+-----------------------------------+-------+
```

- Setting **idle_readonly_transaction_timeout**

  Set **idle_readonly_transaction_timeout** to **5**.

  a. Run the **begin** statement to start a transaction and run a query statement. The following information is displayed.

  ```
  begin;
  Query OK, 0 rows affected (0.00 sec)
  select * from t1;
  +-----+-----+-----+-----+
  | a   | b  | c  | d   |
  +-----+-----+-----+-----+
  | 1   | b  | 303 | d   |
  ```

 Copyright © Huawei Technologies Co., Ltd.

```
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

b. Wait for five seconds and run a query statement again. The following information is displayed.

```
select * from t1;
+-----+-----+-----+-----+
| a   | b  | c  | d   |
+-----+-----+-----+-----+
| 1   | b  | 303 | d   |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
select * from t1;
ERROR 2006(HY000): MySQL server has gone away
```

- Setting **idle_transaction_timeout**, **idle_readonly_transaction_timeout**, and **idle_write_transaction_timeout**

  Set **idle_transaction_timeout** to **10**, **idle_readonly_transaction_timeout** to **0**, and **idle_write_transaction_timeout** to **0**.

```
show variables like '%idle%';
+-----------------------------------+-------+
| Variable_name                     | Value |
+-----------------------------------+-------+
| idle_readonly_transaction_timeout | 0     |
| idle_transaction_timeout          | 10    |
| idle_write_transaction_timeout    | 0     |
+-----------------------------------+-------+
3 rows in set (0.01 sec)
```

  – Read-only transactions

    When **idle_readonly_transaction_timeout** is set to **0**, the **idle_transaction_timeout** parameter takes effect.

    i. Run the **begin** statement to start a transaction and run a statement to query the table data. The following information is displayed.

```
begin;
Query OK, 0 rows affected (0.00 sec)
select * from t1;
+-----+-----+-----+-----+
| a   | b  | c  | d   |
+-----+-----+-----+-----+
| 1   | b  | 43 | d   |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

    ii. Wait for 10 seconds and run a query statement again. The following information is displayed.

```
select * from t1;
ERROR 2006(HY000): MySQL server has gone away
```

  – Read/write transactions

    When **idle_write_transaction_timeout** is set to **0**, the **idle_transaction_timeout** parameter takes effect.

    i. Run the **begin** statement to start a transaction, insert data, and run a query statement within 10 seconds. The following information is displayed.

```
begin;
Query OK, 0 rows affected (0.00 sec)
INSERT INTO t1(a,b,c,d) VALUES (1,'b',FLOOR( 1 + (RAND()*1000)) ,'d');
Query OK, 1 rows affected (0.00 sec)
select * from t1;
+-----+-----+-----+-----+
| a   | b  | c  | d   |
+-----+-----+-----+-----+
| 1   | b  | 425 | d   |
```

 Copyright © Huawei Technologies Co., Ltd.

```
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

ii. Wait for 10 seconds and run a query statement again. The following information is displayed.

```
select * from t1;
ERROR 2006(HY000): MySQL server has gone away
```

iii. Independently run a statement to query the table. If the following information is displayed, the transaction has been rolled back.

```
select * from t1;
Empty set (0.00 sec)
```

- Setting **idle_write_transaction_timeout**

  Set **idle_readonly_transaction_timeout** to **15**.

  a. Run the **begin** statement to start a transaction and then insert a data record. The following information is displayed.

```
begin;
Query OK, 0 rows affected (0.00 sec)
select * from t1;
+-----+-----+-----+-----+
| a   | b | c | d   |
+-----+-----+-----+-----+
| 1   | b | 303 | d   |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

  b. Run a query statement within 15 seconds of the time range specified by **idle_write_transaction_timeout**. The following information is displayed.

```
select * from t1;
+-----+-----+-----+-----+
| a   | b | c | d   |
+-----+-----+-----+-----+
| 1   | b | 987 | d   |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

  c. Wait for 15 seconds and run a query statement again. The following information is displayed.

```
select * from t1;
ERROR 2006(HY000): MySQL server has gone away
```

  d. Reconnect the transaction to the database and run a query statement. If the following information is displayed, the transaction has been rolled back.

```
select * from t1;
Empty set (0.00 sec)
```

 Copyright © Huawei Technologies Co., Ltd.

A Change History

Released On	Description
2021-01-21	This issue is the third official release, which incorporates the following change: Added the kernel version description.
2020-11-20	This issue is the second official release, which incorporates the following change: Added the kernel version updates for MySQL 8.0, MySQL 5.7, and MySQL 5.6.
2020-10-09	This issue is the first official release.

 Copyright © Huawei Technologies Co., Ltd.