

Relational Database Service

Huawei Cloud MySQL Kernel

Issue 01
Date 2022-09-30



Copyright © Huawei Technologies Co., Ltd. 2023. 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



HUAWEI 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://www.huawei.com>

Email: support@huawei.com

Contents

1 Kernel Version Description.....	1
2 Connection Thread Pool.....	3
3 MDL Views.....	7
4 Online Varchar Length Increase.....	11
5 Ending Idle Transactions.....	12
6 DDL Progress Display.....	17

1 Kernel Version Description

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

RDS for MySQL 8.0

Table 1-1 RDS for MySQL 8.0 version description

Date	Description
2022-09-09	<p>The updates in RDS for MySQL 8.0.25 are as follows:</p> <ul style="list-style-type: none">● New features and performance optimized<ul style="list-style-type: none">– Connection per thread was supported for killing sessions.– Constraints on memory were added for the Performance Schema.– The performance of SQL Explorer was optimized based on specific scenarios.– Database performance was optimized in specific scenarios when the value of internal_tmp_mem_storage_engine was set to MEMORY.– The compiler was upgraded to GCC 10.3.● Issues resolved<ul style="list-style-type: none">– Errors caused by writes to a temporary file were resolved.– Unexpected responses to Common Table Expression (CTE) queries were resolved.● Security hardening<ul style="list-style-type: none">– The following security vulnerabilities were fixed: CVE-2021-2417, CVE-2021-2339, CVE-2021-2425, CVE-2021-2426, CVE-2021-2427, CVE-2021-2424, CVE-2021-2383, CVE-2021-2384, and CVE-2021-2410.

RDS for MySQL 5.7

Table 1-2 RDS for MySQL 5.7 version description

Date	Description
2023-06-28	<ul style="list-style-type: none"> ● New features and performance optimized <ul style="list-style-type: none"> - The kernel version was upgraded to 5.7.41. - Compiler security options were added. ● Issues resolved <ul style="list-style-type: none"> - The replication exception that may occur when an index is added to a reference table and a foreign key is added to another table concurrently was rectified. - The replication exception that may occur when a child table is deleted after a foreign key table is deleted was rectified. ● Security hardening <ul style="list-style-type: none"> - The following security vulnerabilities were fixed: CVE-2023-21963, CVE-2022-32221, CVE-2023-21840, CVE-2022-2097, CVE-2022-21617, CVE-2022-21608, CVE-2022-21592, CVE-2022-21589, CVE-2022-1292, CVE-2022-27778, CVE-2018-25032, and CVE-2022-21515.
2022-09-09	<ul style="list-style-type: none"> ● New features and performance optimized <ul style="list-style-type: none"> - The kernel version was upgraded to 5.7.38. - The compiler was upgraded to GCC 10.3. - Connection per thread was supported for killing sessions. - The threshold for slow query logs can be set based on the lock wait duration. - ALT security was hardened. ● Issues resolved <ul style="list-style-type: none"> - Recovery security of crashed XA transactions on the primary instance was enhanced. - Abnormal instance reboot when Database Proxy is enabled was resolved. - Abnormal reboot caused by failed memory request for plugins was resolved. ● Security hardening <ul style="list-style-type: none"> - The following security vulnerabilities were fixed: CVE-2022-21454, CVE-2022-21417, CVE-2022-21427, CVE-2022-21451, CVE-2022-21444, and CVE-2022-21460.

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.

Table 2-2 Thread pool parameters that can be modified

Parameter	Dynamic Parameter	Data Type	Value Range	Description
threadpool_enabled	Yes	boolean	[ON,OFF]	<ul style="list-style-type: none"> 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

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

3 MDL Views

Introduction

MySQL Community Edition cannot obtain table MDLs when performance_schema was disabled. If **Waiting for metadata lock** is displayed, blocking DML or DDL, you may need to reboot DB instances because the association among sessions cannot be identified. This has an impact on 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, RDS for 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    |       |
+-----+-----+-----+-----+-----+-----+
```

Table 3-1 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: <ul style="list-style-type: none"> ● 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: <ul style="list-style-type: none"> ● 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 3-2 MDL view example

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

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      | PENDING    | MDL_EXCLUSIVE      | Table metadata lock |          | test |
| t2     |            |
| 5      | PENDING    | 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.

4 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, copy data to new tables and lock the tables to prevent data writes during the length increase. RDS for MySQL has no limitations on the varchar length and allows you to increase it online.

Constraints

This function is available only in RDS for MySQL 5.6 (kernel version 5.6.46 or later) and 5.7 (kernel version 5.7.27 or later).

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
```

5 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

RDS for 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 |
+-----+-----+
```

Table 5-1 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	<p>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.</p> <p>NOTE</p> <p>The parameters idle_readonly_transaction_timeout and idle_write_transaction_timeout have higher priorities than the parameter idle_transaction_timeout.</p> <ul style="list-style-type: none"> If you set a value for idle_readonly_transaction_timeout or idle_write_transaction_timeout and validate the value, idle_transaction_timeout becomes invalid. If only the parameter idle_transaction_timeout has been set and validated, the value of this parameter is used as the timeout interval for read and write operations on 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.

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

- 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.

- 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 |
+---+---+---+---+
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_write_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)
INSERT INTO t1(a,b,c,d) VALUES (1,'b',FLOOR( 1 + (RAND()*1000)) , 'd');
Query OK, 1 rows affected (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)
```

6 DDL Progress Display

Introduction

DDL operations on large tables are time-consuming. However, MySQL Community Edition does not provide you with any information about the DDL execution phase and progress, which may cause great troubles to you.

To solve this problem, RDS for MySQL launches the DDL progress display feature. You can query the **INFORMATION_SCHEMA.INNODB_ALTER_TABLE_PROGRESS** table to view the execution phase and progress of DDL statements in real time.

Characteristics

Table 6-1 INNODB_ALTER_TABLE_PROGRESS table columns

Column	Description
THREAD_ID	Thread ID
QUERY	ALTER TABLE SQL statements
START_TIME	DDL start time
ELAPSED_TIME	Elapsed time (s)
ALTER_TABLE_STAGE	ALTER TABLE stage events
STAGE_COMPLETED	Completed work at the current stage
STAGE_ESTIMATED	Estimated work at the current stage

In order of occurrence, ALTER TABLE stage events include:

- **stage/innodb/alter table (read PK and internal sort):** Read the primary key.
- **stage/innodb/alter table (merge sort):** Sort by primary key. This process may take a long period of time because temporary files are generated.
- **stage/innodb/alter table (insert):** Insert the sorted data into the table.
- **stage/innodb/alter table (log apply index):** Apply DML logs generated during DDL execution to the created or modified index.

- **stage/innodb/alter table (flush)**: Flush data to the disk.
- **stage/innodb/alter table (log apply table)**: Apply DML logs generated during DDL execution to the created or modified table.
- **stage/innodb/alter table (end)**: Finish the remaining work.