ALIBABA CLOUD

Alibaba Cloud

ApsaraDB for RDS

AliSQL Kernel

Document Version: 20201014

Alibaba Cloud

Legal disclaimer

Document conventions

Style	Description	Example
⚠ Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	⚠ Danger: Resetting will result in the loss of user configuration data.
🔔 Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	🔔 Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.
📢 Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	📢 Notice: If the weight is set to 0, the server no longer receives new requests.
? Note	A note indicates supplemental instructions, best practices, tips, and other content.	? Note: You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings> Network> Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.Release notes of minor AliSQL versions 05
2.Feature 23
  2.1. Thread Pool 23
  2.2. Statement outline 26
  2.3. Sequence Engine 33
  2.4. Returning 37
3.Performance 40
  3.1. Binlog in Redo 40
  3.2. Statement Queue 42
  3.3. Inventory Hint 48
4.Stability 53
  4.1. Statement concurrency control 53
  4.2. Performance Agent 57
  4.3. Purge Large File Asynchronously 65
  4.4. Performance Insight 66
5.Security 73
  5.1. Data Protect 73
  5.2. Recycle bin 74
6.Best practices 79
  6.1. Convert the storage engine of DRDS from InnoDB to X-E... 79
  6.2. DingTalk secures App Store top rank with X-Engine 81
  6.3. Storage engine that processes trillions of Taobao orders 83
  6.4. Best practices for X-Engine testing 85

1.Release notes of minor AliSQL versions

This topic describes the release notes of minor AliSQL versions.

ApsaraDB RDS for MySQL 8.0

20200831

- New features:
  - An option is added to disable parallel scan for the `COUNT (*)` function.
  - Start global transaction identifiers (GTIDs) and end GTIDs are introduced to the mysqlbinlog plug-in.
  - Various log sequence numbers (LSNs) in the redo log are supported.
    - innodb_lsn: the LSN of each record in the redo log.
    - innodb_log_checkpoint_lsn: the LSN of the last checkpoint.
    - innodb_log_write_lsn: the LSN of each record that is written into the redo log.
    - innodb_log_ready_for_write_lsn: the LSN of the last record that is written into the log buffer.
    - innodb_log_flush_lsn: the LSN of each record that is flushed from the redo log to the disk.
    - innodb_log_dirty_pages_added_up_to_lsn: the LSN of each record that logs a page as dirty.
    - innodb_log_oldest_lsn: the LSN of each record that logs an update to a page.
- Performance optimization:
  - The concurrency control (CCL) mechanism is optimized to better determine how transactions can wait and concurrently run.
  - The CCL mechanism is optimized to better prioritize the stored procedures that are to run.
- Bugs fixed:
  - The bug that prevents the recursively called interpreter from checking the memory size is fixed.
  - The bug that prevents you from modifying table definitions when transparent data encryption (TDE) is enabled is fixed.
  - The bug that causes the event scheduler to leak memory is fixed.

20200630

- New features:
  - The faster DDL feature is introduced to provide an optimized buffer pool management mechanism. This mechanism reduces the impact of data definition language (DDL) operations and increases the number of concurrent DDL operations that are allowed. For more information, see Faster DDL.
  - The maximum number of connections that are allowed is increased to 500,000.
- Performance optimization:
  - Thread pools are optimized.

  - The memory allocation mechanism is optimized. You can specify the maximum number of memory resources that are allowed for Performance Schema based on the instance type.
  - SQL log files are no longer detected.
  - TDE is optimized to cache the keys that are provided by Alibaba Cloud Key Management Service (KMS).
  - The status of threads that are managed by the CCL mechanism is modified. For more information, see Statement concurrency control.

- Bugs fixed:
  - The bug that causes the system to consider a semicolon (;) to be a part of the command used to create an outline is fixed.
  - The bug that causes the server to unexpectedly exit in the event of table modifications is fixed.
  - The bug that causes earlier versions to disallow the memory and array keywords supported in later versions is fixed.
  - The bug that causes the system to incorrectly count the number of waits when commands are read from a client is fixed.
  - The bug that causes failures in minor engine version updates is fixed.

20200430

- New features:
  - The Binlog in Redo feature is introduced. This feature writes binary logs into redo log files before the binary logs are written to the disk. This reduces I/O consumption and improves database performance. For more information, see Binlog in Redo.
  - The data protection feature is introduced. This feature supports the customization of security policies that are used to manage the permissions on DROP and TRUNCATE statements. This allows you to avoid data losses that are caused by the unintentional execution of these statements. For more information, see Data Protect.
  - The code for row caching in the X-Engine storage engine is restructured.
  - The XA_RECOVER_ADMIN permission is provided.

- Performance optimization:
  - The code that is used to scan data when operations are performed on a temporary InnoDB table is optimized. This allows the system to scan only dirty pages instead of the entire buffer pool.
  - The global parameter opt_readonly_trans_implicit_commit is renamed as rds_disable_explicit_trans. This ensures compatibility with MySQL 5.6.
  - The SQL Explorer (SQL Audit) feature is optimized, so it does not log upgrades to RDS instances.
  - Memory resources that are consumed by DDL operations on X-Engine tables are reduced.

- Bugs fixed:
  - The bug that causes the sizes of X-Engine tables stored on the disk to be inconsistent with the statistical information in the INFORMATION_SCHEMA schema is fixed.
  - The bug that causes the system to initialize X-Engine logs when the error log file is re-opened is fixed.

20200331

- New feature:

  The `TRUNCATE TABLE` statement is supported. After you execute this statement on a table, this statement moves the table to the recycle bin. Then, this statement creates a table by using the schema of the table that you truncate. For more information, see Recycle bin.
- Performance optimization:
  - The output of Transmission Control Protocol (TCP) errors is disabled by default.
  - The performance of thread pools with the default configuration is improved.
- Bugs fixed:
  - The bug that databases and tables become invalid because the names of partitioned tables are separated with a pound key and a letter p (`#p`) is fixed.
  - The bug that causes the statements managed by the CCL mechanism to be case-sensitive is fixed.
- Changes incorporated: Changes in MySQL 8.0.17 and MySQL 8.0.18 are incorporated. For more information, see Changes in MySQL 8.0.17 and Changes in MySQL 8.0.18.

20200229

- New features:
  - The performance agent feature is introduced. For more information, see Performance Agent. This feature is provided as a MySQL plug-in. It allows you to collect and analyze the performance metrics of an RDS instance.
  - Network round-trip time is introduced to the semi-synchronous mode. This allows you to better understand the performance of an RDS instance.
- Performance optimization:
  - Statement-level CCL is allowed on read-only RDS instances.
  - Outlines are supported for secondary RDS instances.
  - The database proxy feature is enhanced to optimize short-lived connections.
  - The time that is required to execute a PAUSE statement is reduced in various CPU architectures.
  - A memory table is introduced to present the running status of thread pools.
- Bugs fixed:
  - The bug that causes the system to forbid the ppoll function and replace the ppoll function with the poll function in Linux kernels earlier than version 4.9 is fixed.
  - The bug that causes errors when the system invokes the wrap_sm4_encrypt function is fixed.
  - The bug that causes the system to lock global variables when SQL logs are rotated is fixed.
  - The bug that causes errors in restoration inconsistency checks is fixed.
  - The bug that causes incorrect time values in the io_statistics table is fixed.
  - The bug that causes the system to unexpectedly exit when invalid compression algorithms are invoked is fixed.
  - The bug that causes user columns in MySQL 8.0 and MySQL 5.6 to be incompatible is fixed.

20200110

- New feature:

  Three hints are introduced. These hints can be used in SELECT, UPDATE, INSERT, and DELETE statements to commit and roll back transactions at high speeds. This allows you to increase the throughput of your application. For more information, see Inventory Hint.

- Performance optimization:
  - The CCL mechanism is optimized. When an RDS instance is started, CCL queue structures are initialized before CCL rules.
  - The file deletion mechanism is optimized. When you asynchronously delete small files, links to the small files are canceled.
  - The performance of thread pools is optimized. For more information, see Thread Pool.
  - Restoration inconsistency checks are disabled by default.
  - The permissions that are required to configure variables are changed.
    - The user role that is authorized to configure the following variables is changed to standard user:
      - auto_increment_increment
      - auto_increment_offset
      - bulk_insert_buffer_size
      - binlog_rows_query_log_events
    - The user role that is authorized to configure the following variables is changed to superuser or system variable administrator:
      - binlog_format
      - binlog_row_image
      - binlog_direct
      - sql_log_off
      - sql_log_bin

20191225

- New feature:

  The recycle bin feature is introduced. All of the tables that you delete are moved to the recycle bin. You can specify a retention period within which you can retrieve the deleted tables from the recycle bin. For more information, see Recycle bin.

- Performance optimization:
  - The data proxy feature is enhanced to optimize short-lived connections.
  - A dedicated thread is used to serve the maintain user. This allows you to avoid high availability (HA) failures.
  - The locking mechanism is optimized. If an error occurs when binary logs are flushed by using redo logs, ApsaraDB for RDS can explicitly release the lock that is triggered by file synchronization.
  - The deletion of unnecessary TCP error logs is supported.
  - Thread pools are enabled by default.

- Bugs fixed:

  - The bug that causes errors in updates to slow query logs is fixed.
  - The bug that causes an incorrect lock scope is fixed.
  - The bug that causes errors in core dumps when the system invokes the select function for TDE is fixed.

20191115

New feature:

The Statement Queue feature is introduced. This feature allows statements to queue in the same bucket. These statements may be executed on the same resources. For example, these statements are executed on the same row of a table. This reduces overheads from possible conflicts. For more information, see Statement Queue.

20191101

- New features:
  - The SM4 encryption algorithm is supported for TDE. For more information, see Configure TDE for an ApsaraDB RDS for MySQL instance.
  - Data protection for secondary RDS instances is supported. Only the accounts with the SUPER or REPLICATION_SLAVE_ADMIN role have the permissions to insert, delete, and modify data in the slave_master_info, slave_relay_log_info, and slave_worker_info tables.
  - A mechanism is introduced to increase the priorities of auto-increment keys. If a table does not have a primary key or it does not have a unique key without a null value, the auto-increment key without a null value has the highest priority.
  - A mechanism is introduced to prevent the automatic conversion of tables from the MEMORY to MyISAM storage engines. These tables include system tables. These tables also include tables that are invoked by threads in the initializing state.
  - A mechanism is introduced to flush binary log files to the disk before redo log files.
  - A mechanism is introduced to stop the creation of temporary tables on an RDS instance when the RDS instance is locked.
  - The X-Engine storage engine is provided to store transactions based on a log-structured merge (LSM) tree.
- Performance optimization:
  - The thread pool feature is optimized to reduce mutexes. For more information, see Thread Pool.
  - The performance insight feature is optimized to monitor thread pools. For more information, see Performance Insight.
  - Parameter adjustment:
    - `primary_fast_lookup` : a session parameter. Default value: true.
    - `thread_pool_enabled` : a global parameter. Default value: true.

20191015

- New features:
  - The TDE feature is introduced to support real-time I/O encryption and decryption on data files. Data is encrypted before it is written to the disk and decrypted before it is read from the disk to the memory. For more information, see Configure TDE for an ApsaraDB RDS for MySQL instance.

  - The returning feature is introduced. This feature allows data manipulation language (DML) statements to return result sets. In addition, the DBMS_TRANS package is provided for you to use this feature. For more information, see Returning.
  - The forced conversion from the MyISAM or MEMORY storage engine to the InnoDB storage engine is supported. If the global variable **force_mysiam_to_innodb** or **force_memory_to_innodb** is set to **ON**, a table is converted from the MyISAM or MEMORY storage engine to the InnoDB storage engine when the table is created or modified.
  - A mechanism is introduced to forbid standard accounts from performing primary/secondary switchovers. Only privileged accounts have the permissions to perform primary/secondary switchovers.
  - A performance proxy plug-in is provided. This plug-in obtains performance data and saves the data as TXT files to your computer. These files are deleted in a circular manner. Only the latest second-level files are retained.
  - A configurable timeout period is introduced for mutexes in InnoDB: This timeout period can be changed by setting the global variable **innodb_fatal_semaphore_wait_threshold**. The default value of the global variable is 600.
  - Index hint errors can be ignored by setting the global variable **ignore_index_hint_error**. The default value of the global variable is false.
  - The SSL encryption feature can be disabled. For more information, see Configure SSL encryption on an ApsaraDB RDS for MySQL instance.
  - The output of TCP errors is supported. TCP errors in read, read-wait, and write-wait events are returned with their error codes by using end_connection events. In addition, logs with information about the errors are generated.
- Bugs fixed:
  - The bug that prevents a Linux operating system from merging local asynchronous I/O (AIO) requests before linear Read Ahead is triggered is fixed.
  - The bug that prevents the proper collection of table and index statistics is fixed.
  - The bug that prevents the system direct access to the primary key index of a table with a primary key is fixed.

20190915

Bug fixed:

The bug that causes memory leaks when the Cmd_set_current_connection process runs is fixed.

20190816

- New features:
  - The thread pool feature is introduced to separate threads from sessions. If a large number of sessions exist, the system can run a small number of threads to complete the tasks in active sessions. For more information, see Thread Pool.
  - The CCL mechanism is introduced. This mechanism allows you to specify the maximum number of concurrent requests that are allowed. This enables the system to handle traffic bursts, process statements that consume excessive resources, and adapt to changes of SQL models. This also ensures the continuity and stability of your database service. For more information, see Statement concurrency control.
  - The statement outline feature is introduced to support optimizer hints and index hints. These hints are used to stabilize the execution of query plans on an RDS instance. For more information, see Statement outline.

  - The Sequence engine is introduced to simplify the acquisition of sequence values. For more information, see Sequence Engine.
  - The Purge Large File Asynchronously feature is introduced to asynchronously delete files. Before you delete a tablespace, the system renames the files in the tablespace as temporary files. Then, a background thread is started to asynchronously delete the temporary files. For more information, see Purge Large File Asynchronously.
  - The performance insight feature is introduced to support load monitoring, association analysis, and performance optimization at the instance level. This feature allows you to evaluate the loads of an RDS instance. This feature also allows you to locate performance issues to ensure the stability of your database service. For more information, see Performance Insight.
  - An optimized instance locking mechanism is introduced. You can delete tables from an RDS instance by using DROP or TRUNCATE statements even if the RDS instance is locked.
- Bugs fixed:
  - The bug that causes the system to incorrectly calculate file sizes is fixed.
  - The bug that allows irrelevant processes to reuse released memory resources is fixed.
  - The bug that causes a host to exit unexpectedly when the available cache size on the host is 0 is fixed.
  - The bug that causes conflicts between implicit primary keys and CTS statements is fixed.
  - The bug that causes the system to incorrectly log slow queries is fixed.

20190601

- Performance optimization:
  - Metadata locking on logging tables is reduced.
  - The code for termination options is restructured.
- Bugs fixed:
  - The bug that prevents the SQL Explorer (SQL Audit) feature from logging precompiled statements is fixed.
  - The bug that prevents the system from filtering out error logs in logging tables with invalid names is fixed.

ApsaraDB RDS for MySQL 5.7 on RDS Basic or High-availability Edition

20200831

- New features:
  - Changes incorporated: Changes in MySQL 5.7.30 are incorporated. For more information, visit GitHub.
  - An optimized CCL mechanism is introduced to better determine how transactions can wait and concurrently run.
  - Start GTIDs and end GTIDs are introduced to the mysqlbinlog plug-in.

- Various LSNs in the redo log are supported.
    - innodb_lsn: the LSN of each record in the redo log.
    - innodb_log_write_lsn: the LSN of each record that is written into the redo log.
    - innodb_log_checkpoint_lsn: the LSN of the last checkpoint.
    - innodb_log_flushed_lsn: the LSN of each record that is flushed from the redo log to the disk.
    - innodb_log_Pages_flushed: the LSN of each record that logs an update to a page.

- Performance optimization:

  The CCL mechanism is optimized to better prioritize the stored procedures that are to run.
- Bug fixed:

  Some major bugs that cause the server to unexpectedly stop when you shut down the server is fixed.

20200630

- New features:
    - Three hints are introduced to the inventory hint feature. These hints are used in SELECT, UPDATE, INSERT, and DELETE statements to commit and roll back transactions at high speeds. This allows you to increase the throughput of your application. For more information, see Inventory Hint.
    - The CCL mechanism is introduced. This mechanism allows you to specify the maximum number of concurrent requests that are allowed. This enables the system to handle traffic bursts, process statements that consume excessive resources, and adapt to changes of SQL models. This also ensures the continuity and stability of your database service. For more information, see Statement concurrency control.
    - The Statement Queue feature is introduced. This feature allows statements to queue in the same bucket. These statements may be executed on the same resources. For example, these statements are executed on the same row of a table. This reduces overheads from possible conflicts. For more information, see Statement Queue.
    - The statement outline feature is introduced to support optimizer hints and index hints. These hints are used to stabilize the execution of query plans on an RDS instance. For more information, see Statement outline.
    - The faster DDL feature is introduced to provide an optimized buffer pool management mechanism. This mechanism reduces the impact of DDL operations and increases the number of concurrent DDL operations that are allowed. For more information, see Faster DDL.
    - The maximum number of connections that are allowed is increased to 500,000.

- Performance optimization:
    - The `call dbms_admin.show_native_procedure();` command is provided to display all of the procedures on an RDS instance.
    - A new function is provided to delete orphan tables.
    - Thread pools are optimized.
    - Query caching is optimized.
    - The memory allocation mechanism is optimized. You can specify the maximum number of memory resources that are allowed for Performance Schema based on the instance type.

- Bug fixed:

  The bug that causes an audit update thread to enter an infinite loop is fixed.

20200430

- New feature:

  The data protection feature is introduced. This feature supports the customization of security policies that are used to manage the permissions on DROP and TRUNCATE statements. This allows you to avoid data losses that are caused by the unintentional execution of these statements. For more information, see Data Protect.

- Performance optimization:

  Read-write locks are no longer supported in the query cache. The default hash function is changed from LF_hash to murmur3 hash.

- Bugs fixed:

  Two bugs that occur after the system hits the query cache during the execution of transactions at the REPEATABLE_READ isolation level are fixed.

20200331

- New features:
  - The fast query cache is introduced. It is developed by Alibaba Cloud based on the native MySQL query cache. It uses a new design and implementation mechanism to improve query performance. For more information, see Fast query cache.
  - Two metadata locks are introduced from Percona Server 5.7: LOCK TABLES FOR BACKUP (LTFB) and LOCK BINLOG FOR BACKUP (LBFB).
- Performance optimization:
  - Thread pools are optimized to ensure compatibility with earlier MySQL versions.
  - The output of TCP errors is disabled by default.
  - The performance of thread pools with the default configuration is improved.
- Bugs fixed:
  - The bug that causes the system to delete temporary files when you delete large files is fixed.
  - The bug that causes dump threads in thread pools to time out is fixed.
  - The bug that causes the system to incorrectly count the value of the IPK field in the procedure context is fixed.
  - The bug that causes rds_change_user to incur pfs thread leakage and release is fixed.
- Changes incorporated: Changes in MySQL 5.7.28 are incorporated. For more information, visit GitHub.

20200229

- New features:
  - The performance agent feature is introduced. For more information, see Performance Agent. This feature is provided as a MySQL plug-in. It allows you to collect and analyze the performance metrics of an RDS instance.
  - Network round-trip time is introduced to the semi-synchronous mode. This allows you to better understand the performance of an RDS instance.

- Performance optimization:
  - The time that is required to execute a PAUSE statement is reduced in various CPU architectures.
  - The database proxy feature is enhanced to optimize short-lived connections.
  - A memory table is introduced to present the running status of thread pools.
- Bugs fixed:
  - The bug that causes DDL redo logs that are not secure is fixed.
  - The bug that causes incorrect time values in the io_statistics table is fixed.
  - The bug that causes the server to unexpectedly exit in the event of table modifications is fixed.
  - The bugs in MySQL test cases are fixed.

20200110

Performance optimization:

- The file deletion mechanism is optimized. When you asynchronously delete small files, links to the small files are canceled.
- The performance of thread pools is optimized. For more information, see Thread Pool.
- The default value of the thread_pool_enabled parameter is changed to OFF.

20191225

- New feature:

  The management of internal accounts is supported. This allows you to manage user permissions and protect your data.
- Performance optimization:
  - The data proxy feature is enhanced to optimize short-lived connections.
  - A dedicated thread is used to serve the maintain user. This allows you to avoid HA failures.
  - The deletion of unnecessary TCP error logs is supported.
  - Thread pools are optimized.
- Bugs fixed:
  - The bug that causes the mysqld process to unexpectedly exit when the read/write splitting function is enabled is fixed.
  - The bug that causes errors in core dumps when the system uses a keyring is fixed.

20191115

Bug fixed:

The bug that causes the system to display variables in SQL logs that are generated from primary/secondary switchovers is fixed.

20191101

- New features:
  - The SM4 encryption algorithm is supported for TDE. For more information, see Configure TDE for an ApsaraDB RDS for MySQL instance.

  - A mechanism is introduced to allow the system to access the primary index of a table with a primary key.
  - A mechanism is introduced to prevent the automatic conversion of tables from the MEMORY to MyISAM storage engines. These tables include system tables. These tables also include tables that are invoked by threads in the initializing state.
- Performance optimization:
  - The thread pool feature is optimized to reduce mutexes. For more information, see Thread Pool.
  - An SQL log caching mechanism is introduced to increase SQL logging performance.
  - The performance insight feature is optimized to monitor thread pools. For more information, see Performance Insight.
  - The thread pool feature is enabled by default. For more information, see Thread Pool.
- Bugs fixed:
  - The bug that prevents the release of locks on user tables when these tables are being managed or maintained is fixed.
  - More TCP errors are added.

20191015

- New features:
  - The rotation of slow query logs is supported. Every CSV slow query log file is assigned a unique name and a new file. This prevents data losses during the collection of slow query logs. You can run the `show variables like '%rotate_log_table%';` command to check whether the rotation of slow query logs is enabled.
  - A performance proxy plug-in is provided. This plug-in obtains performance data and saves the data as TXT files to your computer. These files are deleted in a circular manner. Only the latest second-level files are retained.
  - The forced conversion from the MEMORY to InnoDB storage engines is supported. If the global variable **rds_force_memory_to_innodb** is set to **ON**, a table is converted from the MEMORY to InnoDB storage engines when the table is created or modified.
  - The keyring-rds plug-in is introduced to TDE. This plug-in allows ApsaraDB for RDS to communicate with the administration system or Alibaba Cloud Key Management Service (KMS).
  - The output of TCP errors is supported. TCP errors in read, read-wait, and write-wait events are returned with their error codes by using end_connection events. In addition, logs with information about the errors are generated.
- Bug fixed:

  The bug that causes Error 1290 in DDL operations is fixed.

20190925

Parameter adjustment:

- The default value of the system variable auto_generate_certs is changed from true to false.
- The global read-only variable auto_detact_certs is introduced. Valid values: true and false. Default value: false. This variable is supported only when code is compiled by using OpenSSL on the server. This variable specifies whether the server automatically searches for SSL

certificate and key files in the data directory.

20190915

New feature:

The thread pool feature is introduced to separate threads from sessions. If a large number of sessions exist, the system can run a small number of threads to complete the tasks in active sessions. For more information, see Thread Pool.

20190815

- New features:
  - The Purge Large File Asynchronously feature is introduced to asynchronously delete files. Before you delete a tablespace, the system renames the files in the tablespace as temporary files. Then, a background thread is started to asynchronously delete the temporary files. For more information, see Purge Large File Asynchronously.
  - The performance insight feature is introduced to support load monitoring, association analysis, and performance optimization at the instance level. This feature allows you to evaluate the loads of an RDS instance. This feature also allows you to locate performance issues to ensure the stability of your database service. For more information, see Performance Insight.
  - An optimized instance locking mechanism is introduced. You can delete tables from an RDS instance by using DROP or TRUNCATE statements even if the RDS instance is locked.
- Bugs fixed:
  - The bug that allows you to set the rds_prepare_begin_id option in the `set rds_current_connection` command is fixed.
  - The bug that prevents the system from updating information about locked accounts is fixed.
  - The bug that allows you to use actual as a keyword in table names is fixed.
  - The bug that causes the overflow of timestamps in slow query logs is fixed.

20190510

New feature: Temporary tables can be created in transactions.

20190319

New feature: Thread IDs can be configured in handshake packets.

20190131

- The upgrade to MySQL 5.7.25 is supported.
- JeMalloc that is used for memory management is disabled.
- The bug that causes the system to incorrectly calculate the value of the internal variable net_lenth_size is fixed.

20181226

- New feature: Dynamic modifications to the system variable binlog-row-event-max-size are supported. This allows you to expedite the replication of tables that do not have a primary key.
- Bug fixed: The bug that prevents the proxy instance of an RDS instance from applying for memory resources is fixed.

20181010

- Implicit primary keys are supported.
- The replication of tables that do not have a primary key between primary and secondary RDS instances is accelerated.
- Native AIO is provided to improve I/O performance.

20180431

New features:

- The RDS High-availability Edition is supported.
- The SQL Audit feature is supported. For more information, see .
- The protection for RDS instances on which snapshot backups are being created is enhanced.

ApsaraDB RDS for MySQL 5.7 on RDS Enterprise Edition

20191128

- New feature:

  The read/write splitting function is introduced.
- Bugs fixed:
  - The bug that causes the system to incorrectly calculate the value of the Second_Behind_Master metric for a follower is fixed.
  - The bug that causes dead locks during the re-execution of table-level parallel replication transactions is fixed.
  - XA-related bugs are fixed.

20191016

- New features:
  - The upgrade from the RDS High-availability Edition to the RDS Enterprise Edition is supported for RDS instances that use local SSDs.
  - The GTID function that is provided by the MySQL Community edition is supported. This function is disabled by default.
  - All of the Alibaba Cloud-proprietary AliSQL features and functions that are released in the RDS Basic and High-availability Editions before the minor version 20190915 are incorporated.
- Bug fixed:

  The bug that causes the system to disable binary logs for reset secondary RDS instances is fixed.

20190909

- New features:
  - The execution of large transactions is accelerated. This applies when the synchronous mode is used to replicate data between primary and secondary RDS instances that run the RDS Enterprise Edition.
  - Binary logs can be dumped from a leader or a follower.
  - The creation of read-only RDS instances is supported.
  - The InnoDB storage engine is used for system tables by default.
- Bugs fixed:

- The bug that invalidates the commands that are run by a follower to delete logs is fixed.
- The bug that causes slave threads to unexpectedly exit when the slave_sql_verify_checksum parameter is set to OFF and the binlog_checksum parameter is set to crc32 is fixed.

20190709

New features:

- The RDS Enterprise Edition is supported.
- The disabling of the semi-sync plug-in is supported.
- Table-level parallel replication and write set-level parallel replication are supported.
- The pk_access module is introduced to expedite queries that are run based on primary keys.
- Thread pools are supported.
- All of the Alibaba Cloud-proprietary AliSQL features and functions that are released for MySQL 5.7 in the RDS Basic and High-availability Editions before the minor version 20190510 are incorporated.

ApsaraDB RDS for MySQL 5.6

20200831

- New features:

  Various LSNs in the redo log are supported.

  - innodb_lsn: the LSN of each record in the redo log.
  - innodb_log_write_lsn: the LSN of each record that is written into the redo log.
  - innodb_log_checkpoint_lsn: the LSN of the last checkpoint.
  - innodb_log_flushed_lsn: the LSN of each record that is flushed from the redo log to the disk.
  - innodb_log_Pages_flushed: the LSN of each record that logs an update to a page.

- Bugs fixed:
  - The bug that causes incorrect SHOW_HA_ROWS enumeration types is fixed.
  - The bug that causes the system to incorrectly count the value of the IPK field in the procedure context is fixed.
  - The bug that causes the server to unexpectedly exit when you query data from the INFORMATION_SCHEMA schema is fixed.
  - The bug that causes an audit update thread to enter an infinite loop is fixed.
  - The bug that prevents secondary RDS instances from reporting data replication latencies is fixed.

20200630

- New features:
  - The performance agent feature is introduced. For more information, see Performance Agent. This feature is provided as a MySQL plug-in. It allows you to collect and analyze the performance metrics of an RDS instance.
  - The maximum number of connections that are allowed is increased to 500,000.

    - The faster DDL feature is introduced to provide an optimized buffer pool management mechanism. This mechanism reduces the impact of DDL operations and increases the number of concurrent DDL operations that are allowed. For more information, see Faster DDL.

- Performance optimization:
    - The global parameter max_execution_time is introduced. If the execution duration of an SQL statement exceeds the value of this parameter, the execution is paused.
    - Thread pools are optimized.
- Bug fixed:

  The bug that causes the system to incorrectly count the number of waits when commands are read from a client is fixed.

20200430

- The data protection feature is introduced. This feature supports the customization of security policies that are used to manage the permissions on DROP and TRUNCATE statements. This allows you to avoid data losses that are caused by the unintentional execution of these statements. For more information, see Data Protect.
- The mdl_info table is provided to store information about metadata locks.
- The bug that causes conflicts when the thread pool and ic_reduce features are both enabled is fixed.

20200331

Performance optimization:

- The performance of thread pools with the default configuration is improved.
- The output of TCP errors is disabled by default.

20200229

- New feature:

  The read/write splitting function is supported for database proxies.
- Performance optimization:
    - Thread pools are optimized.
    - The time that is required to execute a PAUSE statement is reduced in various CPU architectures.
- Bug fixed:

  The bug that causes the system to partially commit XA transactions is fixed.

20200110

- New feature:

  The thread pool feature is introduced to separate threads from sessions. If a large number of sessions exist, the system can run a small number of threads to complete the tasks in active sessions. For more information, see Thread Pool.
- Performance optimization:

  The file deletion mechanism is optimized. When you asynchronously delete small files, links to the small files are canceled.

- Bugs fixed:
  - The bug that causes the system to incorrectly calculate the sleep time of the page cleaner is fixed.
  - The bug that causes the `SELECT @@global.gtid_executed` statement to incur a failover failure is fixed.
  - The bug that causes the IF CLIENT KILLED AFTER ROLLBACK TO SAVEPOINT PREVIOUS STMTS COMMITTED error is fixed.

20191212

Performance optimization:

The deletion of unnecessary TCP error logs is supported.

20191115

Bug fixed:

The bug that causes the overflow of timestamps in slow query logs is fixed.

20191101

Bugs fixed:

- The bug that causes the system to rotate slow query logs when you update common logs is fixed.
- Some display-related bugs are fixed.

20191015

- New features:
  - The rotation of slow query logs is supported. Every CSV slow query log file is assigned a unique name and a new file. This prevents data losses during the collection of slow query logs. You can run the `show variables like '%rotate_log_table%';` command to check whether the rotation of slow query logs is enabled.
  - A new SM4 encryption algorithm is introduced to replace the original SM4 encryption algorithm.
  - The Purge Large File Asynchronously feature is introduced to asynchronously delete files. Before you delete a tablespace, the system renames the files in the tablespace as temporary files. Then, a background thread is started to asynchronously delete the temporary files. For more information, see Purge Large File Asynchronously.
  - The output of TCP errors is supported. TCP errors in read, read-wait, and write-wait events are returned with their error codes by using end_connection events. In addition, logs with information about the errors are generated.
  - An SQL log caching mechanism is introduced to increase SQL logging performance.
- Bugs fixed:
  - The bug that prevents responses to the pstack command when a large number of connections are established is fixed. This is implemented by disabling the pstack command.
  - The bug that causes conflicts between implicit primary keys and `CREATE TABLE AS SELECT` statements is fixed.
  - The bug that prevents the system from deleting the temporary files that are created from binary log files is fixed.

20190815

An optimized instance locking mechanism is introduced. You can delete tables from an RDS instance by using DROP or TRUNCATE statements even if the RDS instance is locked.

20190130

Bugs that compromise database stability are fixed.

20181010

The rocksdb_ddl_commit_in_the_middle parameter is introduced to MyRocks. If this parameter is set to on, some DDL statements call the COMMIT operation when they are executed.

201806** (ApsaraDB RDS for MySQL 5.6.16)

New feature: Microsecond-level time precision is supported for slow query logs.

20180426 (ApsaraDB RDS for MySQL 5.6.16)

- Invisible indexes are supported. For more information, see AliSQL 5.6.32 Release Notes (2017-07-16).
- The bug that causes the system to apply threads on secondary RDS instances is fixed.
- The bug that compromises database performance when updates to partitioned tables are applied on secondary RDS instances is fixed.
- The bug that causes TokuDB to rebuild tables on which ALTER TABLE COMMENT statements are executed is fixed. For more information, see AliSQL 5.6.32 Release Note (2018-05-01).
- The bug that triggers deadlocks when SHOW SLAVE STATUS or SHOW STATUS statements are executed is fixed.

20171205 (ApsaraDB RDS for MySQL 5.6.16)

- The bug that triggers deadlocks when OPTIMIZE TABLE and ONLINE ALTER TABLE statements are executed at the same time is fixed.
- The bug that triggers conflicts between sequences and implicit primary keys is fixed.
- The bug that prevents the proper execution of SHOW CREATE SEQUENCE statements is fixed.
- The bug that causes the system to incorrectly collect statistics on TokuDB tables is fixed.
- The bug that triggers deadlocks when OPTIMIZE statements are executed in parallel on tables is fixed.
- The bug that causes the system to incorrectly record character sets in QUERY_LOG_EVENT is fixed.
- The bug that prevents an RDS instance from stopping due to signal processing issues is fixed. For more information, see AliSQL 5.6.32 Release Notes (2017-10-10).
- The bug that is caused by the execution of RESET MASTER statements is fixed.
- The bug that causes secondary RDS instances to be in a constant waiting state is fixed.
- The bug that causes the database process to unexpectedly exit due to the execution of SHOW CREATE TABLE statements is fixed.

20170927 (ApsaraDB RDS for MySQL 5.6.16)

The bug that causes the system to query tables from TokuDB based on incorrect indexes is fixed.

20170901 (ApsaraDB RDS for MySQL 5.6.16)

- New features:

  - The upgrade of SSL encryption to TLS 1.2 is supported. For more information, see AliSQL 5.6.32 Release Notes (2017-10-10).
  - Sequences are supported.
- Bug fixed: The bug that causes the system to return an incorrect result set for the NOT IN operator is fixed.

20170530 (ApsaraDB RDS for MySQL 5.6.16)

New feature: The privileged account of an RDS instance is granted the permissions to close the connections that are established by all of the standard accounts created on the RDS instance.

20170221 (ApsaraDB RDS for MySQL 5.6.16)

New feature: The read/write splitting function is supported.For more information, see .

ApsaraDB RDS for MySQL 5.5

20181212

The bug that causes the `gettimeofday(2)` function to return an incorrect time value is fixed. The returned time value is used to calculate the timeout period. If the returned time value is incorrect, some operations never time out.

2.Feature

2.1. Thread Pool

ApsaraDB for RDS provides the Thread Pool feature to maximize performance. This feature separates threads from sessions. It allows sessions to share threads and complete more tasks with less threads.

Benefits

By default, each session creates an exclusive thread in MySQL. If a large number of sessions are active, they will compete for resources. In addition, your database system needs to process a heavy workload of thread scheduling, and a large amount of data in the cache becomes invalid. This decreases your database performance.

The thread pool of ApsaraDB for RDS grants priorities to SQL statements based on the statement types. The thread pool also provides a concurrency control mechanism to limit the number of connections. This ensures high database performance in the event of a large number of highly concurrent connections. The benefits of the thread pool are as follows:

- When a large number of threads are running concurrently, the thread pool automatically limits the number of concurrent threads to a proper range. Within this range, your database system processes a moderate workload of thread scheduling, and most of the data in the cache remains valid.
- When a large number of transactions are executed concurrently, the thread pool automatically grants different priorities to SQL statements and transactions. Based on the priorities, the thread pool limits the number of concurrent statements and transactions separately. This mitigates resource competition.
- The thread pool grants high priorities to SQL statements that are used to manage data and ensures that these statements are preferentially executed. This delivers stable execution of operations such as connection establishment, management, and monitoring even if your database system is heavily loaded.
- The thread pool grants low priorities to complicated SQL statements that are used to query data and limits the maximum number of concurrent statements. This prevents a large number of complicated SQL statements from exhausting resources and making the database service unavailable.

Prerequisites

Your RDS instance is running MySQL 5.6, 5.7, or 8.0.

Use the thread pool

The following table describes the parameters of the thread pool. You can configure these parameters in the ApsaraDB for RDS console. For more information, see Reconfigure the parameters of an ApsaraDB RDS for MySQL instance.

Parameter	Description

Parameter	Description
thread_pool_enabled	Specifies whether to enable the Thread Pool feature. Valid values: • ON • OFF Default value: ON. Note • You can enable or disable the Thread Pool feature only by using this parameter. The **thread_handling** parameter has phased out. • Enabling or disabling the Thread Pool feature does not require an instance restart.
thread_pool_size	The number of groups in the thread pool. Default value: 4. Threads in the thread pool are evenly divided into groups and managed by group.
thread_pool_oversubscribe	The number of active threads allowed per group. Default value: 32. A thread is active if it is executing an SQL statement. However, if the SQL statement is in one of the following states, the thread is inactive: • The SQL statement is waiting for disk I/O. • The SQL statement is waiting for the involved transaction to be committed.

Query the status of the thread pool

Run the following command to query the status of the thread pool:

```
show status like "thread_pool%";
```

Example:

```
mysql> show status like "thread_pool%";
+---------------------------+-------+
| Variable_name             | Value |
+---------------------------+-------+
| thread_pool_active_threads | 1    |
| thread_pool_big_threads    | 0    |
| thread_pool_dml_threads    | 0    |
| thread_pool_idle_threads   | 19   |
| thread_pool_qry_threads    | 0    |
| thread_pool_total_threads  | 20   |
| thread_pool_trx_threads    | 0    |
| thread_pool_wait_threads   | 0    |
+---------------------------+-------+
8 rows in set (0.00 sec)
```

The following table describes the parameters that describe the status of the thread pool.

Parameter	Description
thread_pool_active_threads	The number of active threads in the thread pool.
thread_pool_big_threads	The number of threads that are executing complicated SQL statements in the thread pool. Complicated SQL statements contain subqueries, aggregate functions, and clauses such as GROUP BY and LIMIT.
thread_pool_dml_threads	The number of threads that are executing data manipulation language (DML) statements in the thread pool.
thread_pool_idle_threads	The number of idle threads in the thread pool.
thread_pool_qry_threads	The number of threads that are executing simple SQL statements in the thread pool.
thread_pool_total_threads	The total number of threads in the thread pool.
thread_pool_trx_threads	The number of threads that are executing transactions in the thread pool.
thread_pool_wait_threads	The number of threads that are waiting for disk I/O and those that are waiting for transactions to be committed in the thread pool.

Use SysBench to test the thread pool

The following figures show comparisons of performance between business scenarios with the thread pool enabled and disabled. Based on the test results, the thread pool significantly increases your database performance in the event of a large number of highly concurrent sessions.

2.2. Statement outline

Databases may be unstable because the execution plan of SQL statements is constantly changing. Alibaba Cloud provides the statement outline feature to make stable execution plans by using optimizer and index hints. The DBMS_OUTLN package can be installed to use the statement outline feature.

Prerequisites

The RDS instance version is one of the following:

- MySQL 8.0
- MySQL 5.7

Feature design

The statement outline feature supports the following types of hints provided by MySQL 8.0.

- Optimizer hint

  Optimizer hints are classified by scope and object, and are divided into various types, such as global level hint, table level hint, index level hint, and JOIN_ORDER hint. For more information, see Optimizer Hints.

- Index hint

  Index hints are classified by scope and type. For more information, see Index Hints.

Introduction to the outline table

AliSQL uses a system table named outline to store hints. The instance system automatically creates the table when the system is started. You can refer to the following statements that create the outline table.

```
CREATE TABLE `mysql`.`outline` (
  `Id` bigint(20) NOT NULL AUTO_INCREMENT,
  `Schema_name` varchar(64) COLLATE utf8_bin DEFAULT NULL,
  `Digest` varchar(64) COLLATE utf8_bin NOT NULL,
  `Digest_text` longtext COLLATE utf8_bin,
  `Type` enum('IGNORE INDEX','USE INDEX','FORCE INDEX','OPTIMIZER') CHARACTER SET utf8 COLLATE utf8
_general_ci NOT NULL,
  `Scope` enum('','FOR JOIN','FOR ORDER BY','FOR GROUP BY') CHARACTER SET utf8 COLLATE utf8_general
_ci DEFAULT '',
  `State` enum('N','Y') CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL DEFAULT 'Y',
  `Position` bigint(20) NOT NULL,
  `Hint` text COLLATE utf8_bin NOT NULL,
  PRIMARY KEY (`Id`)
) /*! 50100 TABLESPACE `mysql` */ ENGINE=InnoDB
DEFAULT CHARSET=utf8 COLLATE=utf8_bin STATS_PERSISTENT=0 COMMENT='Statement outline'
```

The following table describes the parameters.

Parameter	Description
Id	The ID of the outline table.
Schema_name	The name of the database.
Digest	The 64-byte hash string calculated from **Digest_text** during the hash calculation.
Digest_text	The digest of the SQL statement.
Type	• The hint type of optimizer hints is OPTIMIZER. • The hint type of index hints is USE INDEX, FORCE INDEX, or IGNORE INDEX.
Scope	This parameter is only specified for index hints. Valid values: • FOR GROUP BY • FOR ORDER BY • FOR JOIN An empty string indicates index hints of all types.
State	Specifies whether to enable the statement outline.

Parameter	Description
Position	• For optimizer hints, the Position parameter is the position of the keyword in query blocks because all optimizer hints are applied to query blocks. The value of Position indicates the order of the keyword that is applied by hints. The valid values of Position starts from 1. • For index hints, the Position parameter is the position of the table. The value of Position indicates the order of the table that is applied by hints. The valid value starts from 1.
Hint	• For optimizer hints, Hint indicates an integrated hint string, such as `/*+ MAX_EXECUTION_TIME(1000) */` . • For index hints, Hint indicates a list of index names, such as `ind_1,ind_2` .

Manage the statement outline

AliSQL provides six management interfaces in the DBMS_OUTLN package. They are described as follows:

- add_optimizer_outline

  Adds optimizer hints. The statement is as follows:

  ```
  dbms_outln.add_optimizer_outline('<Schema_name>','<Digest>','<query_block>','<hint>','<query>');
  ```

  Note You can enter either of the Digest or Query SQL statements. If you enter the query statement, DBMS_OUTLN calculates the values of Digest and Digest_text.

  Example:

  ```
  CALL DBMS_OUTLN.add_optimizer_outline("outline_db", '', 1, '/*+ MAX_EXECUTION_TIME(1000) */',
                                  "select * from t1 where id = 1");
  ```

- add_index_outline

  Adds index hints. The statement is as follows:

  ```
  dbms_outln.add_index_outline('<Schema_name>','<Digest>',<Position>,'<Type>','<Hint>','<Scope>','<
  Query>');
  ```

  Note You can enter either of the Digest or Query SQL statements. If you enter the query statement, DBMS_OUTLN calculates the values of Digest and Digest_text.

  Example:

  ```
  call dbms_outln.add_index_outline('outline_db', '', 1, 'USE INDEX', 'ind_1', '',
                                "select * from t1 where t1.col1 =1 and t1.col2 ='xpchild'");
  ```

- preview_outline

  Queries the status of the SQL statement matching the statement outline, which can be used for manual verification. The statement is as follows:

  ```
  dbms_outln.preview_outline('<Schema_name>','<Query>');
  ```

  Example:

  ```
  mysql> call dbms_outln.preview_outline('outline_db', "select * from t1 where t1.col1 =1 and t1.col2 ='x
  pchild'");
  +------------+------------------------------------------------------------------+------------+------------+-------+-
  --------------------+
  | SCHEMA     | DIGEST                                                           | BLOCK_TYPE | BLOCK_NAME | BLOCK | HINT
  |
  +------------+------------------------------------------------------------------+------------+------------+-------+-
  --------------------+
  | outline_db | b4369611be7ab2d27c85897632576a04bc08f50b928a1d735b62d0a140628c4c | TABLE      | t
  1          |     1 | USE INDEX (`ind_1`) |
  +------------+------------------------------------------------------------------+------------+------------+-------+-
  --------------------+
  1 row in set (0.00 sec)
  ```

- show_outline

  Displays the in-memory hit rate of the statement outline. The statement is as follows:

  ```
  dbms_outln.show_outline();
  ```

  Example:

```
mysql> call dbms_outln.show_outline();
+------+-----------+------------------------------------------------------------------+----------+-------+------+
---------------------------------------------------+------+----------+-----------------------------------------
----------------------------------------+
| ID   | SCHEMA     | DIGEST                                                           | TYPE      | SCOPE | POS  | HINT
| HIT  | OVERFLOW | DIGEST_TEXT                                                                          |
+------+-----------+------------------------------------------------------------------+----------+-------+------+
---------------------------------------------------+------+----------+-----------------------------------------
----------------------------------------+
|   33 | outline_db | 36bebc61fce7e32b93926aec3fdd790dad5d895107e2d8d3848d1c60b74bcde6 | OPTIM
IZER |       |    1 | /*+ SET_VAR(foreign_key_checks=OFF) */               |    1 |        0 | SELECT * FROM `t1` WH
ERE `id` = ?                                     |
|   32 | outline_db | 36bebc61fce7e32b93926aec3fdd790dad5d895107e2d8d3848d1c60b74bcde6 | OPTIM
IZER |       |    1 | /*+ MAX_EXECUTION_TIME(1000) */                     |    2 |        0 | SELECT * FROM `t1` WHER
E `id` = ?                                       |
|   34 | outline_db | d4dcef634a4a664518e5fb8a21c6ce9b79fccb44b773e86431eb67840975b649 | OPTIMI
ZER |       |    1 | /*+ BNL(t1,t2) */                                    |    1 |        0 | SELECT `t1` . `id` , `t2` . `id` FROM `t1
` , `t2`                              |
|   35 | outline_db | 5a726a609b6fbfb76bb8f9d2a24af913a2b9d07f015f2ee1f6f2d12dfad72e6f | OPTIMIZ
ER |       |    2 |  /*+ QB_NAME(subq1) */                               |    2 |        0 | SELECT * FROM `t1` WHERE `t1` . `
col1` IN ( SELECT `col1` FROM `t2` )             |
|   36 | outline_db | 5a726a609b6fbfb76bb8f9d2a24af913a2b9d07f015f2ee1f6f2d12dfad72e6f | OPTIMIZ
ER |       |    1 | /*+ SEMIJOIN(@subq1 MATERIALIZATION, DUPSWEEDOUT) */ |    2 |        0 | SELECT * FROM
`t1` WHERE `t1` . `col1` IN ( SELECT `col1` FROM `t2` )              |
|   30 | outline_db | b4369611be7ab2d27c85897632576a04bc08f50b928a1d735b62d0a140628c4c | USE IN
DEX |       |    1 | ind_1                                                |    3 |        0 | SELECT * FROM `t1` WHERE `t1` . `col1` =
? AND `t1` . `col2` = ?                   |
|   31 | outline_db | 33c71541754093f78a1f2108795cfb45f8b15ec5d6bff76884f4461fb7f33419 | USE INDEX
|       |    2 | ind_2                                                |    1 |        0 | SELECT * FROM `t1` , `t2` WHERE `t1` . `col1` =
`t2` . `col1` AND `t2` . `col2` = ? |
+------+-----------+------------------------------------------------------------------+----------+-------+------+
---------------------------------------------------+------+----------+-----------------------------------------
----------------------------------------+
7 rows in set (0.00 sec)
```

The following table describes the HIT and OVERFLOW parameters.

Parameter	Description
HIT	The number of times that the statement outline finds the destination query block or table.

Parameter	Description
OVERFLOW	The number of times that the statement outline does not find the destination query block or table.

- del_outline

  Deletes a statement outline from the memory and the table. The statement is as follows:

```
dbms_outln.del_outline(<Id>);
```

  Example:

```
mysql> call dbms_outln.del_outline(32);
```

> **Note** If the statement outline that you want to delete does not exist, the system displays a corresponding error. You can execute the `SHOW WARNINGS;` statement to view the error message.

```
mysql> call dbms_outln.del_outline(1000);
Query OK, 0 rows affected, 2 warnings (0.00 sec)

mysql> show warnings;
+---------+------+----------------------------------------------+
| Level   | Code | Message                                      |
+---------+------+----------------------------------------------+
| Warning | 7521 | Statement outline 1000 is not found in table |
| Warning | 7521 | Statement outline 1000 is not found in cache |
+---------+------+----------------------------------------------+
2 rows in set (0.00 sec)
```

- flush_outline

  If you modify the statement outline in the outline table, you need to execute the following statement so that the statement outline takes effect again. The statement is as follows:

```
dbms_outln.flush_outline();
```

  Example:

```
mysql> update mysql.outline set Position = 1 where Id = 18;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> call dbms_outln.flush_outline();
Query OK, 0 rows affected (0.01 sec)
```

Feature test

There are two methods to verify whether the statement outline takes effect.

- Use the preview_outline interface.

```
mysql> call dbms_outln.preview_outline('outline_db', "select * from t1 where t1.col1 =1 and t1.col2 ='x
pchild'");
+------------+------------------------------------------------------------------+------------+------------+-------+-
--------------------+
| SCHEMA     | DIGEST                                                           | BLOCK_TYPE | BLOCK_NAME | BLOCK | HINT
|
+------------+------------------------------------------------------------------+------------+------------+-------+-
--------------------+
| outline_db | b4369611be7ab2d27c85897632576a04bc08f50b928a1d735b62d0a140628c4c | TABLE      | t
1          |     1 | USE INDEX (`ind_1`) |
+------------+------------------------------------------------------------------+------------+------------+-------+-
--------------------+
1 row in set (0.01 sec)
```

- Execute the EXPLAIN statement.

```
mysql> explain select * from t1 where t1.col1 =1 and t1.col2 ='xpchild';
+----+-------------+-------+------------+------+---------------+-------+---------+-------+------+----------+-----
--------+
| id | select_type | table | partitions | type | possible_keys | key   | key_len | ref   | rows | filtered | Extra
|
+----+-------------+-------+------------+------+---------------+-------+---------+-------+------+----------+-----
--------+
|  1 | SIMPLE      | t1    | NULL       | ref  | ind_1         | ind_1 | 5       | const |    1 |   100.00 | Using where |
+----+-------------+-------+------------+------+---------------+-------+---------+-------+------+----------+-----
--------+
1 row in set, 1 warning (0.00 sec)

mysql> show warnings;
+-------+------+-----------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
----------------------------+
| Level | Code | Message
|
+-------+------+-----------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
----------------------------+
| Note  | 1003 | /* select#1 */ select `outline_db`.`t1`.`id` AS `id`,`outline_db`.`t1`.`col1` AS `col1`,`outl
ine_db`.`t1`.`col2` AS `col2` from `outline_db`.`t1` USE INDEX (`ind_1`) where ((`outline_db`.`t1`.`col1
` = 1) and (`outline_db`.`t1`.`col2` = 'xpchild')) |
+-------+------+-----------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
----------------------------+
1 row in set (0.00 sec)
```

2.3. Sequence Engine

AliSQL provides the Sequence Engine feature. This feature allows you to use a Sequence engine on your ApsaraDB for RDS instance to simplify the generation of sequence values.

Introduction

Unique, monotonically increasing sequence values are commonly required for the primary keys in a single-node persistent database system, for the globally unique identifiers (GUIDs) in a distributed persistent database system, and for the idempotence among multiple persistent database systems. Each database system may have its unique way to ensure unique sequence values. For example, MySQL provides the AUTO_INCREMENT attribute, and Oracle and SQL Server provide the SEQUENCE attribute.

However, in MySQL databases, it is inconvenient to encapsulate unique sequence values such as dates or user names by using the AUTO_INCREMENT attribute. To make the generation of unique sequence values easier, the following alternative solutions are provided:

- Use an application or a proxy to generate sequence values. However, in this case, the states of the sequence values are sent to the application. This makes scaling more complicated.
- Use a simulated table to generate sequence values. However, middleware must be provided to encapsulate and simplify the logic that is used to obtain the generated sequence values.

A Sequence engine is compatible with various database engines and simplifies the generation of sequence values.

A Sequence engine is compatible with various storage engines used with MySQL. However, the underlying persistent data is still stored by using existing storage engines such as InnoDB and MyISAM. This ensures compatibility with third-party tools such as XtraBackup. Therefore, a Sequence engine is merely a logical engine.

A Sequence engine uses Sequence Handler to access sequence objects. This allows you to increase the value of a sequence by using the NEXTVAL operator and manage the cached data. The data is passed to the underlying base table engine to support data access.

Prerequisites

Your RDS instance is running one of the following database engine versions:

- MySQL 5.6
- MySQL 8.0

Limits

- You cannot perform subqueries or JOIN queries in a Sequence engine.
- You can use the `SHOW CREATE TABLE` or `SHOW CREATE SEQUENCE` statement to access a sequence. However, you cannot use the `SHOW CREATE SEQUENCE` statement to access a regular table.
- You cannot specify a Sequence engine during table creation. If you want to specify a Sequence engine for a table, you must execute the statement described in the "Create a sequence" section.

Create a sequence

To create a sequence, execute the following statement:

```
CREATE SEQUENCE [IF NOT EXISTS] schema.sequence_name
   [START WITH <constant>]
   [MINVALUE <constant>]
   [MAXVALUE <constant>]
   [INCREMENT BY <constant>]
   [CACHE <constant> | NOCACHE]
   [CYCLE | NOCYCLE]
  ;
```

The following table describes the parameters that you need to configure.

Parameter	Description
START	The start value of the sequence.
MINVALUE	The minimum value of the sequence.
MAXVALUE	The maximum value of the sequence. Note If the sequence is specified as NOCYCLE, the following error is reported when the maximum value is reached: `ERROR HY000: Sequence 'db.seq' has been run out.`
INCREMENT BY	The increment at which the value of the sequence increases.
CACHE/NOCACHE	The size of the cache. You can set a larger cache size for better performance. However, if your RDS instance is restarted, sequence values stored in the cache will be lost.
CYCLE/NOCYCLE	Specifies whether the value of the sequence restarts from the specified MINVALUE after reaching the maximum value. Valid values: • CYCLE: The value of the sequence will restart from the specified MINVALUE after reaching the maximum value. • NOCYCLE: The value of the sequence will not restart from the specified MINVALUE after reaching the maximum value.

Example:

```
create sequence s
    start with 1
    minvalue 1
    maxvalue 9999999
    increment by 1
    cache 20
    cycle;
```

If you want to use the mysqldump program to back up your RDS instance, you can create a sequence table and insert an initial row into the sequence table. Example:

```
CREATE SEQUENCE schema.sequence_name (
  `currval` bigint(21) NOT NULL COMMENT 'current value',
  `nextval` bigint(21) NOT NULL COMMENT 'next value',
  `minvalue` bigint(21) NOT NULL COMMENT 'min value',
  `maxvalue` bigint(21) NOT NULL COMMENT 'max value',
  `start` bigint(21) NOT NULL COMMENT 'start value',
  `increment` bigint(21) NOT NULL COMMENT 'increment value',
  `cache` bigint(21) NOT NULL COMMENT 'cache size',
  `cycle` bigint(21) NOT NULL COMMENT 'cycle state',
  `round` bigint(21) NOT NULL COMMENT 'already how many round'
) ENGINE=InnoDB DEFAULT CHARSET=latin1

INSERT INTO schema.sequence_name VALUES(0,0,1,9223372036854775807,1,1,10000,1,0);
COMMIT;
```

Introduction to sequence tables

Sequences are stored in the tables that are created based on the default storage engine. If you query the sequences that you created, the system returns the tables that are created based on the default storage engine. Example:

```
SHOW CREATE [TABLE|SEQUENCE] schema.sequence_name;

CREATE SEQUENCE schema.sequence_name (
  `currval` bigint(21) NOT NULL COMMENT 'current value',
  `nextval` bigint(21) NOT NULL COMMENT 'next value',
  `minvalue` bigint(21) NOT NULL COMMENT 'min value',
  `maxvalue` bigint(21) NOT NULL COMMENT 'max value',
  `start` bigint(21) NOT NULL COMMENT 'start value',
  `increment` bigint(21) NOT NULL COMMENT 'increment value',
  `cache` bigint(21) NOT NULL COMMENT 'cache size',
  `cycle` bigint(21) NOT NULL COMMENT 'cycle state',
  `round` bigint(21) NOT NULL COMMENT 'already how many round'
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Statements supported

A Sequence engine supports the following statements:

```
 SELECT [nextval | currval | *] FROM seq;
 SELECT nextval(seq),currval(seq);
 SELECT seq.currval, seq.nextval from dual;
```

2.4. Returning

This topic describes the Returning feature of AliSQL. This feature enables data manipulation language (DML) statements to return result sets and provides the DBMS_TRANS package for you to track the execution of DML statements.

Context

The execution results of MySQL statements are divided into three types: result sets, OK packets, and ERR packets. An OK or ERR packet contains attributes such as the number of affected and the number of scanned records. However, the execution of a DML statement (INSERT, UPDATE, or DELETE) is often followed by the execution of the SELECT statement to query current records. In such cases, the Returning feature enables the server to respond to the client only once by combining the execution results of the two statements into a result set.

Prerequisites

Your RDS instance is running MySQL 8.0.

Syntax

```
DBMS_TRANS.returning(<Field_list>,<Statement>);
```

The following table describes the parameters that you need to configure.

Parameter	Description
Field_list	The fields to return. If you enter more than one field, separate them with commas (,). Native fields and wildcards (*) in the specified table are supported. However, operations such as calculation and aggregation are not supported.
Statement	The DML statement to execute. Only the INSERT, UPDATE, and DELETE statements are supported.

Precautions

`dbms_trans.returning()` is not a transactional statement. It inherits the context of the specified transaction based on the DML statement that you want to execute. To terminate the transaction, you must explicitly commit it or roll it back.

INSERT Returning

The server returns the records that were inserted into the specified table by using the INSERT statement.

Example:

```
CREATE TABLE `t` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `col1` int(11) NOT NULL DEFAULT '1',
  `col2` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB;

mysql> call dbms_trans.returning("*", "insert into t(id) values(NULL),(NULL)");
+----+------+---------------------+
| id | col1 | col2                |
+----+------+---------------------+
|  1 |    1 | 2019-09-03 10:39:05 |
|  2 |    1 | 2019-09-03 10:39:05 |
+----+------+---------------------+
2 rows in set (0.01 sec)
```

Note

- If you do not specify the Field_list parameter, the server returns an OK or ERR packet.

```
mysql> call dbms_trans.returning("", "insert into t(id) values(NULL),(NULL)");
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from t;
+----+------+---------------------+
| id | col1 | col2                |
+----+------+---------------------+
|  1 |    1 | 2019-09-03 10:40:55 |
|  2 |    1 | 2019-09-03 10:40:55 |
|  3 |    1 | 2019-09-03 10:41:06 |
|  4 |    1 | 2019-09-03 10:41:06 |
+----+------+---------------------+
4 rows in set (0.00 sec)
```

- The Returning feature only supports statements that are similar to `INSERT VALUES`. It does not support statements such as `CREATE AS` and `INSERT SELECT`.

```
mysql> call dbms_trans.returning("", "insert into t select * from t");
ERROR 7527 (HY000): Statement didn't support RETURNING clause
```

UPDATE Returning

The server returns the records that were updated in the specified table by the using UPDATE statement.

Example:

```
mysql> call dbms_trans.returning("id, col1, col2", "update t set col1 = 2 where id >2");
+----+------+---------------------+
| id | col1 | col2                |
+----+------+---------------------+
|  3 |    2 | 2019-09-03 10:41:06 |
|  4 |    2 | 2019-09-03 10:41:06 |
+----+------+---------------------+
2 rows in set (0.01 sec)
```

Note The Returning feature does not allow the UPDATE statement to be executed on more than one table.

DELETE Returning

The server returns the records that were deleted from the specified table by using the DELETE statement.

Example:

```
mysql> call dbms_trans.returning("id, col1, col2", "delete from t where id < 3");
+----+------+---------------------+
| id | col1 | col2                |
+----+------+---------------------+
|  1 |    1 | 2019-09-03 10:40:55 |
|  2 |    1 | 2019-09-03 10:40:55 |
+----+------+---------------------+
2 rows in set (0.00 sec)
```

3.Performance

3.1. Binlog in Redo

The Binlog in Redo function synchronously writes binary logs to the redo log file when a transaction is committed. This reduces operations on disks and improves database performance.

Prerequisites

Your RDS instance runs MySQL 8.0 (with a kernel version of 20200430 or later).

Context

To ensure data security in crucial MySQL business scenarios, the system stores both binary and redo logs when a transaction is committed. Both the following parameters must be set to 1:

```
sync_binlog = 1;
innodb_flush_log_at_trx_commit = 1;
```

Each time a transaction is committed, the system performs two I/O operations. One is to write the binary logs to disks, and the other is to write the redo logs to disks. Although Group Commit is enabled for binary logs, the system must still wait for the two I/O operations to complete. This affects the efficiency of transaction processing, especially when standard or enhanced SSDs are used. The performance of I/O merging is based on the number of concurrent transactions that are committed at the same time. When the number of concurrent transactions is small, the performance is low. For example, when a small number of write transactions are committed, the system response is slow.

To increase the efficiency of committing transactions, AliSQL provides the Binlog in Redo function. You can enable the function by setting the `persist_binlog_to_redo` parameter to on. When a transaction is committed, the system synchronously writes binary logs to the redo log file and stores only the redo log file to disks. This reduces I/O consumption. The binary log files are then asynchronously stored to disks by using a separate thread at regular intervals. If a restart operation is triggered upon an exception, the system uses the binary logs in the redo log file to complement the binary log files. In this way, the database performance improves and the system response is faster. Also, the number of times that the binary log files are stored is reduced. This significantly relieves the pressure on the file system while increasing performance. This pressure can be resulted from the calls of the fsync functions that are triggered by file updates in real time. The fsync function synchronizes files to disks.

Binlog in Redo does not change the format of binary logs. Replication and third-party tools that are based on binary logs are not affected.

Parameters

- persist_binlog_to_redo

  The switch that is used to enable or disable the Binlog in Redo function. This parameter is a global system variable. Valid values: on and off. The parameter change immediately takes effect. You do not need to restart your RDS instance.

Note If you want to enable Binlog in Redo, you only need to set the `persist_binlog_to_redo` parameter to on. You do not need to modify the settings of other parameters. The setting `sync_binlog = 1` automatically becomes invalid.

- sync_binlog_interval

  The interval at which binary logs are asynchronously stored. This parameter is a global system variable. It takes effect only when the `persist_binlog_to_redo` parameter is set to on. Default value: 50. Unit: milliseconds (ms). In normal cases, the default value is recommended. The parameter change immediately takes effect. You do not need to restart your RDS instance.

Stress testing

- Test environment
  - Application server: an Alibaba Cloud ECS instance
  - RDS instance type: 32 CPU cores, 64 GB of memory, and enhanced SSDs
  - RDS edition: High-availability Edition with asynchronous data replication
- Test cases

  Sysbench provides the following test cases:
  - oltp_update_non_index
  - oltp_insert
  - oltp_write_only
- Test results
  - oltp_update_non_index

    After Binlog in Redo is enabled, the queries per second (QPS) significantly increases and the latency is low when the number of concurrent queries is small.
  - oltp_insert

    After Binlog in Redo is enabled, the QPS significantly increases and the latency is low when the number of concurrent queries is small.
  - oltp_write_only

    After Binlog in Redo is enabled, the QPS slightly increases and the latency is low when the number of concurrent queries is small.

- Number of times that the fsync function is called for binary logs

  After Binlog in Redo is enabled, the number of times is significantly reduced.

Test conclusion

- oltp_update_non_index and oltp_insert test single-statement transactions, and the transactions are committed on a frequent basis. oltp_write_only tests multi-statement transactions, and the transactions are committed on a less frequent basis. This type of transaction contains two UPDATE statements, one DELETE statement, and one INSERT statement. Performance improvement in oltp_update_non_index and oltp_insert is more notable than that in oltp_write_only.
- When the number of concurrent transactions is less than 256, Binlog in Redo significantly improves database performance and reduces latency. In most scenarios, Binlog in Redo provides significant benefits.
- Binlog in Redo significantly reduces the number of times that the fsync function is called for binary logs. This improves the performance of the file system.

3.2. Statement Queue

The Statement Queue feature of AliSQL allows statements to queue in the same bucket. These statements may be executed on the same resources. For example, these statements are executed on the same row of a table. This reduces overheads from possible conflicts.

Context

During the execution of concurrent statements, the MySQL server and engine are likely to conflict with each other in a number of serial operations. Take transactional lock conflicts triggered by data manipulation language (DML) statements as an example. The InnoDB storage engine supports resource locking accurate to rows. If you execute a number of DML statements concurrently on a row, serious conflicts may occur. The overall throughput of your database system decreases in proportion with the number of concurrent statements. The Statement Queue feature reduces overheads from these conflicts and increases the performance of your database system.

Prerequisites

The RDS instance version is one of the following:

- MySQL 8.0
- MySQL 5.7

Benefits

AliSQL executes UPDATE statements concurrently on a single row four times faster than native MySQL.

Variables

AliSQL provides two variables that are used to define the bucket quantity and size of a statement queue:

- ccl_queue_bucket_count: the number of buckets allowed in the statement queue. Valid values: 1 to 64. Default value: 4.

- ccl_queue_bucket_size: the number of concurrent statements allowed per bucket. Valid values: 1 to 4096. Default value: 64.

> **Note** You can reconfigure the variables in the ApsaraDB for RDS console. For more information, see Reconfigure the parameters of an ApsaraDB RDS for MySQL instance.

Syntaxes

AliSQL supports two hint syntaxes:

- ccl_queue_value

  AliSQL uses a hash algorithm to determine the bucket into which each statement is placed based on the value of a specified field.

  Syntax:

  ```
  /*+ ccl_queue_value([int | string)] */
  ```

  Example:

  ```
  update /*+ ccl_queue_value(1) */ t set c=c+1 where id = 1;


  update /*+ ccl_queue_value('xpchild') */ t set c=c+1 where name = 'xpchild';
  ```

- ccl_queue_field

  AliSQL uses a hash algorithm to determine the bucket into which each statement is placed based on the value of the field that is specified in the WHERE clause.

  Syntax:

  ```
  /*+ ccl_queue_field(string) */
  ```

  Example:

  ```
  update /*+ ccl_queue_field("id") */ t set c=c+1 where id = 1 and name = 'xpchild';
  ```

  > **Note** In the ccl_queue_field hint, the WHERE clause only supports binary operators on raw fields. These raw fields have not been altered by using functions or computation operations. In addition, the right operand of such a binary operator must be a number or string.

Functions

AliSQL provides two functions that are used to query the status of a statement queue:

- dbms_ccl.show_ccl_queue()

  This function is used to query the status of the current statement queue.

```
mysql> call dbms_ccl.show_ccl_queue();
+------+-------+-------------------+---------+---------+----------+
| ID   | TYPE  | CONCURRENCY_COUNT | MATCHED | RUNNING | WAITTING |
+------+-------+-------------------+---------+---------+----------+
|    1 | QUEUE |                64 |       1 |       0 |        0 |
|    2 | QUEUE |                64 |   40744 |      65 |        6 |
|    3 | QUEUE |                64 |       0 |       0 |        0 |
|    4 | QUEUE |                64 |       0 |       0 |        0 |
+------+-------+-------------------+---------+---------+----------+
4 rows in set (0.01 sec)
```

The following table describes the parameters in this function.

Parameter	Description
CONCURRENCY_COUNT	The maximum number of concurrent queries allowed.
MATCHED	The total number of rules matched.
RUNNING	The number of statements that are being executed concurrently.
WAITTING	The number of statements that are waiting in queue.

- dbms_ccl.flush_ccl_queue()

  This function is used to delete data about the statement queue from the memory and query the status of the statement queue.

```
mysql> call dbms_ccl.flush_ccl_queue();
Query OK, 0 rows affected (0.00 sec)

mysql> call dbms_ccl.show_ccl_queue();
+------+-------+-------------------+---------+---------+----------+
| ID   | TYPE  | CONCURRENCY_COUNT | MATCHED | RUNNING | WAITTING |
+------+-------+-------------------+---------+---------+----------+
|    1 | QUEUE |                64 |       0 |       0 |        0 |
|    2 | QUEUE |                64 |       0 |       0 |        0 |
|    3 | QUEUE |                64 |       0 |       0 |        0 |
|    4 | QUEUE |                64 |       0 |       0 |        0 |
+------+-------+-------------------+---------+---------+----------+
4 rows in set (0.00 sec)
```

Practices

Statement Queue can work with Statement outline to support online updates of your application code. In the following example, SysBench is used to execute the update_non_index.lua script:

- Test environment
  - Schema

    ```
    CREATE TABLE `sbtest1` (
      `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
      `k` int(10) unsigned NOT NULL DEFAULT '0',
      `c` char(120) NOT NULL DEFAULT '',
      `pad` char(60) NOT NULL DEFAULT '',
      PRIMARY KEY (`id`),
      KEY `k_1` (`k`)
    ) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8 MAX_ROWS=1000000;
    ```

  - Statement

    ```
    UPDATE sbtest1 SET c='xpchild' WHERE id=0;
    ```

  - Script

    ```
    ./sysbench
    --mysql-host= {$ip}
    --mysql-port= {$port}
    --mysql-db=test
    --test=./sysbench/share/sysbench/update_non_index.lua
    --oltp-tables-count=1
    --oltp_table_size=1
    --num-threads=128
    --mysql-user=u0
    ```

- Procedure
  1. Create a statement outline in online mode.

     ```
     mysql> CALL DBMS_OUTLN.add_optimizer_outline('test', '', 1,
                                           ' /*+ ccl_queue_field("id") */ ',
                           "UPDATE sbtest1 SET c='xpchild' WHERE id=0");
     Query OK, 0 rows affected (0.01 sec)
     ```

  2. View the statement outline that you created.

```
mysql> call dbms_outln.show_outline();
+------+--------+------------------------------------------------------------------+-----------+-------+------
+-------------------------------+------+----------+------------------------------------------+
| ID   | SCHEMA | DIGEST                                                           | TYPE      | SCOPE | POS  | HINT
| HIT  | OVERFLOW | DIGEST_TEXT                              |
+------+--------+------------------------------------------------------------------+-----------+-------+------
+-------------------------------+------+----------+------------------------------------------+
|    1 | test   | 7b945614749e541e0600753367884acff5df7e7ee2f5fb0af5ea58897910f023 | OPTIMIZE
R |       |    1 |  /*+ ccl_queue_field("id") */  |    0 |        0 | UPDATE `sbtest1` SET `c` = ? WHERE `id` = ?
|
+------+--------+------------------------------------------------------------------+-----------+-------+------
+-------------------------------+------+----------+------------------------------------------+
1 row in set (0.00 sec)
```

iii. Verify that the statement outline has taken effect.

```
mysql> explain UPDATE sbtest1 SET c='xpchild' WHERE id=0;
+----+-------------+---------+------------+-------+---------------+---------+---------+-------+------+------
----+-------------+
| id | select_type | table   | partitions | type  | possible_keys | key     | key_len | ref   | rows | filtere
d | Extra       |
+----+-------------+---------+------------+-------+---------------+---------+---------+-------+------+------
----+-------------+
|  1 | UPDATE      | sbtest1 | NULL       | range | PRIMARY       | PRIMARY | 4       | const |    1 |   100.00 |
Using where |
+----+-------------+---------+------------+-------+---------------+---------+---------+-------+------+------
----+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> show warnings;
+-------+------+--------------------------------------------------------------------------------------------
----------------------------+
| Level | Code | Message                                                                                    |
+-------+------+--------------------------------------------------------------------------------------------
----------------------------+
| Note  | 1003 | update /*+ CCL_QUEUE_FIELD('id') */ `test`.`sbtest1` set `test`.`sbtest1`.`c` = 'xpc
hild' where (`test`.`sbtest1`.`id` = 0) |
+-------+------+--------------------------------------------------------------------------------------------
----------------------------+
1 row in set (0.00 sec)
```

iv. Query the status of the statement queue used.

```
mysql> call dbms_ccl.show_ccl_queue();
+------+-------+-------------------+---------+---------+----------+
| ID   | TYPE  | CONCURRENCY_COUNT | MATCHED | RUNNING | WAITTING |
+------+-------+-------------------+---------+---------+----------+
|    1 | QUEUE |                64 |       0 |       0 |        0 |
|    2 | QUEUE |                64 |       0 |       0 |        0 |
|    3 | QUEUE |                64 |       0 |       0 |        0 |
|    4 | QUEUE |                64 |       0 |       0 |        0 |
+------+-------+-------------------+---------+---------+----------+
4 rows in set (0.00 sec)
```

v. Start the test.

```
sysbench
--mysql-host= {$ip}
--mysql-port= {$port}
--mysql-db=test
--test=./sysbench/share/sysbench/update_non_index.lua
--oltp-tables-count=1
--oltp_table_size=1
--num-threads=128
--mysql-user=u0
```

vi. View the test result.

```
mysql> call dbms_ccl.show_ccl_queue();
+------+-------+-------------------+---------+---------+----------+
| ID   | TYPE  | CONCURRENCY_COUNT | MATCHED | RUNNING | WAITTING |
+------+-------+-------------------+---------+---------+----------+
|    1 | QUEUE |                64 |   10996 |      63 |        4 |
|    2 | QUEUE |                64 |       0 |       0 |        0 |
|    3 | QUEUE |                64 |       0 |       0 |        0 |
|    4 | QUEUE |                64 |       0 |       0 |        0 |
+------+-------+-------------------+---------+---------+----------+
4 rows in set (0.03 sec)

mysql> call dbms_outln.show_outline();
+------+--------+-----------+-----------+-------+------+-------------------------------+--------+----------+
--------------------------------------------+
| ID   | SCHEMA | DIGEST    | TYPE      | SCOPE | POS  | HINT                          | HIT    | OVERFLOW | DIGES
T_TEXT                              |
+------+--------+-----------+-----------+-------+------+-------------------------------+--------+----------+
--------------------------------------------+
|    1 | test   | xxxxxxxxx | OPTIMIZER |       |    1 |  /*+ ccl_queue_field("id") */ | 115795 |        0 | UPDA
TE `sbtest1` SET `c` = ? WHERE `id` = ? |
+------+--------+-----------+-----------+-------+------+-------------------------------+--------+----------+
--------------------------------------------+
1 row in set (0.00 sec)
```

Note Based on the query results, the statement outline is hit 115,795 times, the statement queue is hit 10,996 times, a total of 63 statements are being executed concurrently, and four statements are waiting in queue.

3.3. Inventory Hint

This topic describes the Inventory Hint feature provided by AliSQL. This feature can work with the Returning and Statement Queue features to commit and roll back transactions rapidly.

Background information

In business scenarios such as seckilling, inventory reduction is a common task model that requires high concurrency and serialization. In this model, AliSQL uses queues and transactional hints to control concurrency and commit or roll back transactions. This increases the throughput of your business.

Prerequisites

The RDS instance version is one of the following:

- MySQL 8.0

- MySQL 5.7
- MySQL 5.6

Syntax

The following three hints are introduced to specify tables in SELECT, UPDATE, INSERT, and DELETE statements.

- COMMIT_ON_SUCCESS and ROLLBACK_ON_FAIL

  These are two transactional hints.

  - COMMIT_ON_SUCCESS: specifies to commit the transaction if the execution of the statement to which this hint is applied succeeds.
  - ROLLBACK_ON_FAIL: specifies to roll the transaction back if the execution of the statement to which this hint is applied fails.

  Syntax:

  ```
  /*+ COMMIT_ON_SUCCESS */
  /*+ ROLLBACK_ON_FAIL */
  ```

  Example:

  ```
  UPDATE /*+ COMMIT_ON_SUCCESS ROLLBACK_ON_FAIL */ T
  SET c = c - 1
  WHERE id = 1;
  ```

- TARGET_AFFECT_ROW(NUMBER)

  This is a conditional hint. After you apply it to a statement, the execution of the statement succeeds only when the number of affected rows is the same as the number specified in this hint.

  Syntax:

  ```
  /*+ TARGET_AFFECT_ROW(NUMBER) */
  ```

  Example:

  ```
  UPDATE /*+ TARGET_AFFECT_ROW(1) */ T
  SET c = c - 1
  WHERE id = 1;
  ```

Precautions

- The transactional hints do not support the autocommit mode. If you use a transactional hint in a statement with the autocommit mode, an error is reported. Example:

```
mysql> UPDATE /*+ commit_on_success rollback_on_fail target_affect_row(1) */ t
   -> SET col1 = col1 + 1
   -> WHERE id = 1;
ERROR 7531 (HY000): Inventory transactinal hints didn't allowed in autocommit mode
```

- Transactional hints cannot be used in substatements. If you use a transactional hint in a substatement, an error is reported. Example:

```
 mysql> CREATE TRIGGER tri_1
   -> BEFORE INSERT ON t
   -> FOR EACH ROW
   -> BEGIN
   -> INSERT /*+ commit_on_success */ INTO t1 VALUES (1);
   -> end//

mysql> INSERT INTO t VALUES (2, 1);
ERROR HY000: Inventory transactional hints didn't alllowed in stored procedure
```

- The conditional hint cannot be used in a SELECT or EXPLAIN statement. If you use the conditional hint in a SELECT or EXPLAIN statement, an error is reported. Example:

```
mysql> EXPLAIN UPDATE /*+ commit_on_success rollback_on_fail target_affect_row(1) */ t
   -> SET col1 = col1 + 1
   -> WHERE id = 1;
ERROR 7532 (HY000): Inventory conditional hints didn't match with result
```

Note You can specify an invalid number in the TARGET_AFFECT_ROW hint and check whether the system reports errors:

```
mysql> EXPLAIN UPDATE /*+ commit_on_success rollback_on_fail target_affect_row(-1) */ t
    -> SET col1 = col1 + 1
    -> WHERE id = 1;
+----+-------------+-------+------------+-------+---------------+---------+---------+-------+------+--------
--+-------------+
| id | select_type | table | partitions | type  | possible_keys | key     | key_len | ref   | rows | filtered |
Extra       |
+----+-------------+-------+------------+-------+---------------+---------+---------+-------+------+--------
--+-------------+
|  1 | UPDATE      | t     | NULL       | range | PRIMARY       | PRIMARY | 4       | const |    1 |   100.00 | Using
where |
+----+-------------+-------+------------+-------+---------------+---------+---------+-------+------+--------
--+-------------+
1 row in set, 2 warnings (0.00 sec)

mysql> show warnings;
+---------+------+---------------------------------------------------------------------------------------------------
------------------------------------------+
| Level   | Code | Message
|
+---------+------+---------------------------------------------------------------------------------------------------
------------------------------------------+
| Warning | 1064 | Optimizer hint syntax error near '-1) */ t set col1=col1+1 where id =1' at line 1
|
| Note    | 1003 | update /*+ COMMIT_ON_SUCCESS ROLLBACK_ON_FAIL */ `test`.`t` set `test`.`t`.`c
ol1` = (`test`.`t`.`col1` + 1) where (`test`.`t`.`id` = 1) |
+---------+------+---------------------------------------------------------------------------------------------------
------------------------------------------+
2 rows in set (0.00 sec)
```

Work with Returning

You can use Inventory Hint with Returning for the system to return real-time result sets. Example:

```
mysql> CALL dbms_trans.returning("*", "update /*+ commit_on_success rollback_on_fail target_affect_r
ow(1) */ t
                                        set col1=col1+1 where id=1");
+----+------+
| id | col1 |
+----+------+
|  1 |   13 |
+----+------+
1 row in set (0.00 sec)

mysql> CALL dbms_trans.returning("*", "insert /*+ commit_on_success rollback_on_fail target_affect_ro
w(1) */ into
                                        t values(10,10)");
+----+------+
| id | col1 |
+----+------+
| 10 |   10 |
+----+------+
1 row in set (0.01 sec)
```

Work with Statement Queue

You can use Inventory Hint with Statement Queue for the system to queue statements. Example:

```
mysql> UPDATE /*+ ccl_queue_field(id) commit_on_success rollback_on_fail target_affect_row(1) */ t
    -> SET col1 = col1 + 1
    -> WHERE id = 1;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE /*+ ccl_queue_value(1) commit_on_success rollback_on_fail target_affect_row(1) */ t
    -> SET col1 = col1 + 1
    -> WHERE id = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

4.Stability

4.1. Statement concurrency control

Alibaba Cloud provides the concurrency control (CCL) feature to ensure the stability of ApsaraDB RDS MySQL instances in case of unexpected request traffic, resource-consuming statements, and SQL access model changes. The DBMS_CCL package can be installed to use the CCL feature.

Prerequisites

The RDS instance version is one of the following:

- MySQL 8.0
- MySQL 5.7

Precautions

- CCL operations only affect the current instance because no binlogs are generated. For example, CCL operations performed on the primary instance are not synchronized to the secondary instance, read-only instance, or disaster recovery instance.
- CCL includes a timeout mechanism which resolves transaction deadlocks caused by DML statements. The waiting threads also respond to the transaction timeout and kill threads to prevent deadlocks.

Feature design

The CCL provides features based on the following dimensions:

- SQL command

  The types of SQL statements, such as SELECT, UPDATE, INSERT, and DELETE.
- Object

  The objects managed by SQL statements, such as tables and views.
- keywords

  The keywords of SQL statements.

Create a CCL table

AliSQL uses a system table named concurrency_control to store CCL rules. The instance system automatically creates the table when the system is started. You can refer to the following statements that create the concurrency_control table.

```
CREATE TABLE `concurrency_control` (
  `Id` bigint(20) NOT NULL AUTO_INCREMENT,
  `Type` enum('SELECT','UPDATE','INSERT','DELETE') NOT NULL DEFAULT 'SELECT',
  `Schema_name` varchar(64) COLLATE utf8_bin DEFAULT NULL,
  `Table_name` varchar(64) COLLATE utf8_bin DEFAULT NULL,
  `Concurrency_count` bigint(20) DEFAULT NULL,
  `Keywords` text COLLATE utf8_bin,
  `State` enum('N','Y') NOT NULL DEFAULT 'Y',
  `Ordered` enum('N','Y') NOT NULL DEFAULT 'N',
  PRIMARY KEY (`Id`)
) /*! 50100 TABLESPACE `mysql` */ ENGINE=InnoDB
DEFAULT CHARSET=utf8 COLLATE=utf8_bin
STATS_PERSISTENT=0 COMMENT='Concurrency control'
```

Parameter	Description
Id	The ID of the CCL rule.
Type	The type of the SQL statement.
Schema_name	The name of the database.
Table_name	The name of the table in the database.
Concurrency_count	The number of concurrent threads.
Keywords	The keyword. Multiple keywords are separated by semicolons (;).
State	Specifies whether to enable the CCL rule.
Ordered	Specifies whether to match multiple keywords in sequence.

Manage CCL rules

AliSQL provides four management interfaces in the DBMS_CCL package. They are described as follows:

- add_ccl_rule

  Use the following statement to create a rule.

  ```
  dbms_ccl.add_ccl_rule('<Type>','<Schema_name>','<Table_name>',<Concurrency_count>,'<Keywords>
  ');
  ```

  Example:

  The number of concurrent threads of the SELECT statement is 10.

  ```
  mysql> call dbms_ccl.add_ccl_rule('SELECT', '', '', 10, '');
  ```

The number of concurrent threads of the SELECT statement is 20, and the keyword of the statement is key1.

```
mysql> call dbms_ccl.add_ccl_rule('SELECT', '', '', 20, 'key1');
```

The number of concurrent threads of the SELECT statement in the test.t table is 20.

```
mysql> call dbms_ccl.add_ccl_rule('SELECT', 'test', 't', 20, '');
```

Note The rule with a larger Id has higher priority.

- del_ccl_rule

  Use the following statement to delete a rule.

  ```
  dbms_ccl.del_ccl_rule(<Id>);
  ```

  Example:

  Delete the CCL rule whose ID is 15.

  ```
  mysql> call dbms_ccl.del_ccl_rule(15);
  ```

  Note If the CCL rule that you want to delete does not exist, the system displays an error. You can execute the `SHOW WARNINGS;` statement to view the error message.

  ```
  mysql> call dbms_ccl.del_ccl_rule(100);
   Query OK, 0 rows affected, 2 warnings (0.00 sec)

  mysql> show warnings;
  +---------+------+----------------------------------------------------+
  | Level   | Code | Message                                            |
  +---------+------+----------------------------------------------------+
  | Warning | 7514 | Concurrency control rule 100 is not found in table |
  | Warning | 7514 | Concurrency control rule 100 is not found in cache |
  +---------+------+----------------------------------------------------+
  ```

- show_ccl_rule

  Use the following statement to view the enabled rules in the memory.

  ```
  dbms_ccl.show_ccl_rule();
  ```

  Example:

```
mysql> call dbms_ccl.show_ccl_rule();
+------+--------+--------+-------+-------+-------+-------------------+---------+---------+----------+----------+
| ID   | TYPE   | SCHEMA | TABLE | STATE | ORDER | CONCURRENCY_COUNT | MATCHED | RUNNING | WAITTI
NG | KEYWORDS |
+------+--------+--------+-------+-------+-------+-------------------+---------+---------+----------+----------+
|   17 | SELECT | test   | t     | Y     | N     |                30 |       0 |       0 |        0 |          |
|   16 | SELECT |        |       | Y     | N     |                20 |       0 |       0 |        0 | key1     |
|   18 | SELECT |        |       | Y     | N     |                10 |       0 |       0 |        0 |          |
+------+--------+--------+-------+-------+-------+-------------------+---------+---------+----------+----------+
```

The following table describes the MATCHED, RUNNING, and WAITTING parameters.

Parameter	Description
MATCHED	The number of times the rule is matched.
RUNNING	The number of concurrent threads under the rule.
WAITTING	The number of threads to be run under the rule.

- flush_ccl_rule

  If you modify the rules in the concurrency_control table, you must execute the following statement to validate the rules again.

```
dbms_ccl.flush_ccl_rule();
```

  Example:

```
mysql> update mysql.concurrency_control set CONCURRENCY_COUNT = 15 where Id = 18;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> call dbms_ccl.flush_ccl_rule();
Query OK, 0 rows affected (0.00 sec)
```

Feature test

- Test rules

  Execute the following statements to create the rules for three dimensions.

```
call dbms_ccl.add_ccl_rule('SELECT', 'test', 'sbtest1', 3, '');  // The SELECT statement manages the sbt
est1 table and the number of concurrent threads is 3.
call dbms_ccl.add_ccl_rule('SELECT', '', '', 2, 'sbtest2');      // The keyword of the SELECT statement is s
btest2 and the number of concurrent threads is 2.
 call dbms_ccl.add_ccl_rule('SELECT', '', '', 2, '');            // The number of concurrent threads of the SELE
CT statement is 2.
```

- Test scenarios

  Use sysbench to test in the following scenarios:

  - 64 threads
  - 4 tables
  - select.lua

- Test results

  Execute the following statement to view the number of concurrent threads under the rules.

```
mysql> call dbms_ccl.show_ccl_rule();
+------+--------+--------+---------+-------+-------+-------------------+---------+---------+----------+----------
+
| ID   | TYPE   | SCHEMA | TABLE   | STATE | ORDER | CONCURRENCY_COUNT | MATCHED | RUNNING | WAIT
TING | KEYWORDS |
+------+--------+--------+---------+-------+-------+-------------------+---------+---------+----------+----------
+
|   20 | SELECT | test   | sbtest1 | Y     | N     |                 3 |     389 |       3 |        9 |          |
|   21 | SELECT |        |         | Y     | N     |                 2 |     375 |       2 |       14 | sbtest2  |
|   22 | SELECT |        |         | Y     | N     |                 2 |     519 |       2 |       34 |          |
+------+--------+--------+---------+-------+-------+-------------------+---------+---------+----------+----------
+
3 rows in set (0.00 sec)
```

  The numbers displayed in the **RUNNING** column are the same as the numbers specified when you create the rules.

4.2. Performance Agent

This topic describes the Performance Agent feature provided by AliSQL as a plug-in to collect statistics of performance data on ApsaraDB RDS for MySQL instances.

Background information

Performance Agent adds a memory table named PERF_STATISTICS to the information_schema system database. This table stores the performance data generated over a recent period of time. You can query performance data from this table.

Prerequisites

The RDS instance runs one of the following database engine versions:

- MySQL 8.0 (The kernel version of the RDS instance is 20200229 or later)
- MySQL 5.7 (The kernel version of the RDS instance is 20200229 or later)

Note For information about updating the kernel version, see Upgrade the minor engine version of an ApsaraDB RDS for MySQL instance.

Parameters

The following table describes the parameters you must configure for Performance Agent. For more information, see Reconfigure parameters for an RDS MySQL instance.

Parameter	Description
performance_agent_enabled	Specifies whether to enable the Performance Agent feature. Valid values: ON | OFF. Default value: ON.
performance_agent_interval	The interval at which you want to collect performance data. Unit: seconds. Default value: 1.
performance_agent_perfstat_volume_size	The maximum number of data records that are allowed in the PERF_STATISTICS memory table. Default value: 3600. If you set the performance_agent_interval parameter to 1, the system retains the performance data generated within the last hour.

Schema

The schema of the PERF_STATISTICS memory table is as follows:

```
CREATE TEMPORARY TABLE `PERF_STATISTICS` (
  `TIME` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `PROCS_MEM_USAGE` double NOT NULL DEFAULT '0',
  `PROCS_CPU_RATIO` double NOT NULL DEFAULT '0',
  `PROCS_IOPS` double NOT NULL DEFAULT '0',
  `PROCS_IO_READ_BYTES` bigint(21) NOT NULL DEFAULT '0',
  `PROCS_IO_WRITE_BYTES` bigint(21) NOT NULL DEFAULT '0',
  `MYSQL_CONN_ABORT` int(11) NOT NULL DEFAULT '0',
  `MYSQL_CONN_CREATED` int(11) NOT NULL DEFAULT '0',
  `MYSQL_USER_CONN_COUNT` int(11) NOT NULL DEFAULT '0',
  `MYSQL_CONN_RUNNING` int(11) NOT NULL DEFAULT '0',
  `MYSQL_LOCK_IMMEDIATE` int(11) NOT NULL DEFAULT '0',
  `MYSQL_LOCK_WAITED` int(11) NOT NULL DEFAULT '0',
  `MYSQL_COM_INSERT` int(11) NOT NULL DEFAULT '0',
  `MYSQL_COM_UPDATE` int(11) NOT NULL DEFAULT '0',
  `MYSQL_COM_DELETE` int(11) NOT NULL DEFAULT '0',
  `MYSQL_COM_SELECT` int(11) NOT NULL DEFAULT '0',
  `MYSQL_COM_COMMIT` int(11) NOT NULL DEFAULT '0',
  `MYSQL_COM_ROLLBACK` int(11) NOT NULL DEFAULT '0',
  `MYSQL_COM_PREPARE` int(11) NOT NULL DEFAULT '0',
  `MYSQL_LONG_QUERY` int(11) NOT NULL DEFAULT '0',
  `MYSQL_TCACHE_GET` bigint(21) NOT NULL DEFAULT '0',
  `MYSQL_TCACHE_MISS` bigint(21) NOT NULL DEFAULT '0',
  `MYSQL_TMPFILE_CREATED` int(11) NOT NULL DEFAULT '0',
  `MYSQL_TMP_TABLES` int(11) NOT NULL DEFAULT '0',
  `MYSQL_TMP_DISKTABLES` int(11) NOT NULL DEFAULT '0',
  `MYSQL_SORT_MERGE` int(11) NOT NULL DEFAULT '0',
  `MYSQL_SORT_ROWS` int(11) NOT NULL DEFAULT '0',
  `MYSQL_BYTES_RECEIVED` bigint(21) NOT NULL DEFAULT '0',
  `MYSQL_BYTES_SENT` bigint(21) NOT NULL DEFAULT '0',
  `MYSQL_BINLOG_OFFSET` int(11) NOT NULL DEFAULT '0',
  `MYSQL_IOLOG_OFFSET` int(11) NOT NULL DEFAULT '0',
  `MYSQL_RELAYLOG_OFFSET` int(11) NOT NULL DEFAULT '0',
  `EXTRA` json NOT NULL DEFAULT 'null'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Column	Description
TIME	The time when the data record was generated. The time is in the yyyy-MM-dd HH:mm:ss format.

Column	Description
PROCS_MEM_USAGE	The amount of physical memory occupied by the data record. Unit: bytes.
PROCS_CPU_RATIO	The CPU utilization of the data record.
PROCS_IOPS	The number of I/O operations that the system invoked.
PROCS_IO_READ_BYTES	The amount of data that was read by I/O operations. Unit: bytes.
PROCS_IO_WRITE_BYTES	The amount of data that was written by I/O operations. Unit: bytes.
MYSQL_CONN_ABORT	The number of disconnected connections.
MYSQL_CONN_CREATED	The number of new connections.
MYSQL_USER_CONN_COUNT	The total number of connections.
MYSQL_CONN_RUNNING	The number of active connections.
MYSQL_LOCK_IMMEDIATE	The number of locks held by the data record.
MYSQL_LOCK_WAITED	The number of locks for which the data record waited.
MYSQL_COM_INSERT	The number of statements executed to insert data.
MYSQL_COM_UPDATE	The number of statements executed to update data.
MYSQL_COM_DELETE	The number of statements executed to delete data.
MYSQL_COM_SELECT	The number of statements executed to query data.
MYSQL_COM_COMMIT	The number of transactions explicitly committed.
MYSQL_COM_ROLLBACK	The number of transactions rolled back.
MYSQL_COM_PREPARE	The number of statements that were pre-processed.
MYSQL_LONG_QUERY	The number of slow queries.
MYSQL_TCACHE_GET	The number of cache hits.
MYSQL_TCACHE_MISS	The number of cache misses.
MYSQL_TMPFILE_CREATED	The number of temporary files created.
MYSQL_TMP_TABLES	The number of temporary tables created.
MYSQL_TMP_DISKTABLES	The number of temporary disk tables created.
MYSQL_SORT_MERGE	The number of times that data was merged and sorted.

Column	Description
MYSQL_SORT_ROWS	The number of rows sorted.
MYSQL_BYTES_RECEIVED	The amount of data received. Unit: bytes.
MYSQL_BYTES_SENT	The amount of data sent. Unit: bytes.
MYSQL_BINLOG_OFFSET	The size of the binary log file generated. Unit: bytes.
MYSQL_IOLOG_OFFSET	The size of the binary log file sent from the primary instance. Unit: bytes.
MYSQL_RELAYLOG_OFFSET	The size of the binary log file sent from the secondary instance. Unit: bytes.
EXTRA	The statistics information about InnoDB. The EXTRA parameter consists of multiple fields in the JSON format. For more information, see Fields in the EXTRA parameter. **Note** The values of the metrics in the InnoDB statistics information are the same as the values obtained by executing the `SHOW STATUS` statement.

Fields in the EXTRA parameter

Field	Description
INNODB_TRX_CNT	The number of transactions.
INNODB_DATA_READ	The amount of data read. Unit: bytes.
INNODB_IBUF_SIZE	The number of pages merged.
INNODB_LOG_WAITS	The number of times that InnoDB waited to write logs.
INNODB_MAX_PURGE	The number of transactions deleted.
INNODB_N_WAITING	The number of locks for which InnoDB waited.
INNODB_ROWS_READ	The number of rows read.
INNODB_LOG_WRITES	The number of times that logs were written by InnoDB.
INNODB_IBUF_MERGES	The number of times that data was merged by InnoDB.
INNODB_DATA_WRITTEN	The amount of data written. Unit: bytes.
INNODB_DBLWR_WRITES	The number of double write operations.
INNODB_IBUF_SEGSIZE	The size of data inserted into the buffer.

Field	Description
INNODB_ROWS_DELETED	The number of rows deleted.
INNODB_ROWS_UPDATED	The number of rows updated.
INNODB_COMMIT_TRXCNT	The number of transactions committed.
INNODB_IBUF_FREELIST	The length of the idle list.
INNODB_MYSQL_TRX_CNT	The number of MySQL transactions.
INNODB_ROWS_INSERTED	The number of rows inserted.
INNODB_ACTIVE_TRX_CNT	The number of active transactions.
INNODB_OS_LOG_WRITTEN	The amount of log data written. Unit: bytes.
INNODB_ACTIVE_VIEW_CNT	The number of active views.
INNODB_RSEG_HISTORY_LEN	The length of the TRX_RSEG_HISTORY table.
INNODB_AVG_COMMIT_TRXTIME	The average time taken to commit a transaction.
INNODB_MAX_COMMIT_TRXTIME	The maximum time taken to commit a transaction.
INNODB_DBLWR_PAGES_WRITTEN	The number of writes completed by double write operations.

Procedure

- Query the system table to obtain performance data.
  - Query the CPU utilization and memory usage within the last 30 seconds. Example:

```
MySQL> select TIME, PROCS_MEM_USAGE, PROCS_CPU_RATIO from information_schema.PERF_STAT
ISTICS order by time DESC limit 30;
+---------------------+-----------------+-----------------+
| TIME                | PROCS_MEM_USAGE | PROCS_CPU_RATIO |
+---------------------+-----------------+-----------------+
| 2020-02-27 11:15:36 |       857812992 |           18.55 |
| 2020-02-27 11:15:35 |       857808896 |           18.54 |
| 2020-02-27 11:15:34 |       857268224 |           19.64 |
| 2020-02-27 11:15:33 |       857268224 |           21.06 |
| 2020-02-27 11:15:32 |       857264128 |           20.39 |
| 2020-02-27 11:15:31 |       857272320 |           20.32 |
| 2020-02-27 11:15:30 |       857272320 |           21.35 |
| 2020-02-27 11:15:29 |       857272320 |            28.8 |
| 2020-02-27 11:15:28 |       857268224 |           29.08 |
| 2020-02-27 11:15:27 |       857268224 |           26.92 |
| 2020-02-27 11:15:26 |       857268224 |           23.84 |
| 2020-02-27 11:15:25 |       857264128 |           13.76 |
| 2020-02-27 11:15:24 |       857264128 |           15.12 |
| 2020-02-27 11:15:23 |       857264128 |           14.76 |
| 2020-02-27 11:15:22 |       857264128 |           15.38 |
| 2020-02-27 11:15:21 |       857260032 |           13.23 |
| 2020-02-27 11:15:20 |       857260032 |           12.75 |
| 2020-02-27 11:15:19 |       857260032 |           12.17 |
| 2020-02-27 11:15:18 |       857255936 |           13.22 |
| 2020-02-27 11:15:17 |       857255936 |           20.51 |
| 2020-02-27 11:15:16 |       857255936 |           28.74 |
| 2020-02-27 11:15:15 |       857251840 |           29.85 |
| 2020-02-27 11:15:14 |       857251840 |           29.31 |
| 2020-02-27 11:15:13 |       856981504 |           28.85 |
| 2020-02-27 11:15:12 |       856981504 |           29.19 |
| 2020-02-27 11:15:11 |       856977408 |           29.12 |
| 2020-02-27 11:15:10 |       856977408 |           29.32 |
| 2020-02-27 11:15:09 |       856977408 |            29.2 |
| 2020-02-27 11:15:08 |       856973312 |           29.36 |
| 2020-02-27 11:15:07 |       856973312 |           28.79 |
+---------------------+-----------------+-----------------+
30 rows in set (0.08 sec)
```

- Query the rows that are read and written by InnoDB within the last 30 seconds. Example:

```
MySQL> select TIME, EXTRA->'$.INNODB_ROWS_READ', EXTRA->'$.INNODB_ROWS_INSERTED' from inf
ormation_schema.PERF_STATISTICS order by time DESC limit 30;
+---------------------+----------------------------+--------------------------------+
| TIME                | EXTRA->'$.INNODB_ROWS_READ' | EXTRA->'$.INNODB_ROWS_INSERTED' |
+---------------------+----------------------------+--------------------------------+
| 2020-02-27 11:22:17 | 39209                      | 0                              |
| 2020-02-27 11:22:16 | 36098                      | 0                              |
| 2020-02-27 11:22:15 | 38035                      | 0                              |
| 2020-02-27 11:22:14 | 37384                      | 0                              |
| 2020-02-27 11:22:13 | 38336                      | 0                              |
| 2020-02-27 11:22:12 | 33946                      | 0                              |
| 2020-02-27 11:22:11 | 36301                      | 0                              |
| 2020-02-27 11:22:10 | 36835                      | 0                              |
| 2020-02-27 11:22:09 | 36900                      | 0                              |
| 2020-02-27 11:22:08 | 36402                      | 0                              |
| 2020-02-27 11:22:07 | 39672                      | 0                              |
| 2020-02-27 11:22:06 | 39316                      | 0                              |
| 2020-02-27 11:22:05 | 37830                      | 0                              |
| 2020-02-27 11:22:04 | 36396                      | 0                              |
| 2020-02-27 11:22:03 | 34820                      | 0                              |
| 2020-02-27 11:22:02 | 37350                      | 0                              |
| 2020-02-27 11:22:01 | 39463                      | 0                              |
| 2020-02-27 11:22:00 | 38419                      | 0                              |
| 2020-02-27 11:21:59 | 37673                      | 0                              |
| 2020-02-27 11:21:58 | 35117                      | 0                              |
| 2020-02-27 11:21:57 | 36140                      | 0                              |
| 2020-02-27 11:21:56 | 37592                      | 0                              |
| 2020-02-27 11:21:55 | 39765                      | 0                              |
| 2020-02-27 11:21:54 | 35553                      | 0                              |
| 2020-02-27 11:21:53 | 35882                      | 0                              |
| 2020-02-27 11:21:52 | 37061                      | 0                              |
| 2020-02-27 11:21:51 | 40699                      | 0                              |
| 2020-02-27 11:21:50 | 39608                      | 0                              |
| 2020-02-27 11:21:49 | 39317                      | 0                              |
| 2020-02-27 11:21:48 | 37413                      | 0                              |
+---------------------+----------------------------+--------------------------------+
30 rows in set (0.08 sec)
```

- Connect to a performance monitoring platform to monitor your database performance in real time. For example, connect to Grafana.

4.3. Purge Large File Asynchronously

This topic describes how to use the Purge Large File Asynchronously function to delete files from an ApsaraDB for RDS instance running AliSQL. This function is designed to ensure database stability by deleting large files asynchronously.

Context

If your ApsaraDB for RDS instance runs the InnoDB storage engine, directly deleting large files from the instance compromises the stability of your POSIX file system. As a result, InnoDB starts a background thread to delete large files asynchronously. InnoDB renames data files housing tablespaces to identify them as temporary files before starting to delete the tablespaces asynchronously.

Note AliSQL ensures the atomicity of Data Definition Language (DDL) statements by deleting log files.

Procedure

1. View the global variable settings of your RDS instance, as shown in the following example:

```
mysql> SHOW GLOBAL VARIABLES LIKE '%data_file_purge%';
 +----------------------------------------+-------+
 | Variable_name                          | Value |
 +----------------------------------------+-------+
 | innodb_data_file_purge                 | ON    |
 | innodb_data_file_purge_all_at_shutdown | OFF   |
 | innodb_data_file_purge_dir             |       |
 | innodb_data_file_purge_immediate       | OFF   |
 | innodb_data_file_purge_interval        | 100   |
 | innodb_data_file_purge_max_size        | 128   |
 | innodb_print_data_file_purge_process   | OFF   |
 +----------------------------------------+-------+
```

   The following table describes these variables.

Variable	Description
innodb_data_file_purge	Specifies whether to enable the Purge Large File Asynchronously function.
innodb_data_file_purge_all_at_shutdown	Specifies whether to delete all files when the host server of your RDS instance is shut down.

Variable	Description
innodb_data_file_purge_dir	The directory for stored temporary files.
innodb_data_file_purge_immediate	Specifies whether to revoke data file links, but not to delete them.
innodb_data_file_purge_interval	The intervals at which files are deleted. Unit: ms.
innodb_data_file_purge_max_size	The maximum size of a single file that can be deleted. Unit: MB.
innodb_print_data_file_purge_process	Specifies whether to display the file deletion process.

Note We recommend that you set the following variables to the values provided in the example:

```
set global INNODB_DATA_FILE_PURGE = on;
set global INNODB_DATA_FILE_PURGE_INTERVAL = 100;
set global INNODB_DATA_FILE_PURGE_MAX_SIZE = 128;
```

2. Run the following command to view the file deletion progress:

```
select * from information_schema.build_current_task
```

4.4. Performance Insight

This topic describes how to use the Performance Insight function for load monitoring, association analysis, and optimizing performance. This function helps you quickly evaluate the loads of your ApsaraDB for RDS instance and locate performance problems to ensure database stability.

Prerequisites

- Your RDS instance runs one of the following database engine versions:
  - MySQL 8.0
  - MySQL 5.7
- The kernel version of your RDS instance is 20190915 or later.

  Note Log on to the ApsaraDB for RDS console, find the target RDS instance, and navigate to the **Basic Information** page. Then in the **Configuration Information** section, check whether the **Upgrade Kernel Version** button is available. If the button is available, click it to view the kernel version of your RDS instance. If the button is not available, you are already using the latest kernel version. For more information, see Upgrade the minor engine version of an ApsaraDB RDS for MySQL instance.

Overview

The Performance Insight function consists of the following two parts:

- Object Statistics

  Object Statistics queries statistics from indexes and the following two tables:

  - TABLE_STATISTICS: records rows with read and modified data.
  - INDEX_STATISTICS: records rows with data read from indexes.

- Performance Point

  Performance Point collects performance details of your RDS instance. Using these details, you can quantify the overheads of SQL statements faster and more accurately. Performance Point measures database performance using the following three dimensions:

  - CPU: includes but is not limited to the total time spent executing an SQL statement and the time spent by CPU executing an SQL statement.
  - Lock: includes the time occupied by locks such as metadata locks on the server, storage transaction locks, mutual exclusions (mutexes) (in debugging mode only), and readers-writer locks.
  - I/O: includes the time taken to perform operations such as reading and writing data files, writing log files, reading binary logs, reading redo logs, and asynchronously reading redo logs.

Use Object Statistics

1. Check that the values of the OPT_TABLESTAT and OPT_INDEXSTAT parameters are ON. Example:

   ```
   mysql> show variables like "opt_%_stat";
    +---------------+-------+
    | Variable_name | Value |
    +---------------+-------+
    | opt_indexstat | ON    |
    | opt_tablestat | ON    |
    +---------------+-------+
   ```

   Note If these parameters cannot be found or their values are not ON, check that your RDS instance is running MySQL 5.7.

2. Query the TABLE_STATISTICS or INDEX_STATISTICS table in the information_schema database to obtain table or index statistics. Examples:

```
mysql> select * from TABLE_STATISTICS limit 10;
 +--------------+--------------+-----------+-------------+-----------------------+---------------+-------------
-+--------------+
 | TABLE_SCHEMA | TABLE_NAME   | ROWS_READ | ROWS_CHANGED | ROWS_CHANGED_X_INDEXES | R
OWS_INSERTED | ROWS_DELETED | ROWS_UPDATED |
 +--------------+--------------+-----------+-------------+-----------------------+---------------+-------------
-+--------------+
 | mysql        | db           |         2 |           0 |                     0 |             0 |            0 |            0 |
 | mysql        | engine_cost  |         2 |           0 |                     0 |             0 |            0 |            0 |
 | mysql        | proxies_priv |         1 |           0 |                     0 |             0 |            0 |            0 |
 | mysql        | server_cost  |         6 |           0 |                     0 |             0 |            0 |            0 |
 | mysql        | tables_priv  |         2 |           0 |                     0 |             0 |            0 |            0 |
 | mysql        | user         |         7 |           0 |                     0 |             0 |            0 |            0 |
 | test         | sbtest1      |      1686 |         142 |                   184 |           112 |           12 |           18 |
 | test         | sbtest10     |      1806 |         125 |                   150 |           105 |            5 |           15 |
 | test         | sbtest100    |      1623 |         141 |                   182 |           110 |           10 |           21 |
 | test         | sbtest11     |      1254 |         136 |                   172 |           110 |           10 |           16 |
 +--------------+--------------+-----------+-------------+-----------------------+---------------+-------------
-+--------------+

 mysql> select * from INDEX_STATISTICS limit 10;
 +--------------+--------------+------------+-----------+
 | TABLE_SCHEMA | TABLE_NAME   | INDEX_NAME | ROWS_READ |
 +--------------+--------------+------------+-----------+
 | mysql        | db           | PRIMARY    |         2 |
 | mysql        | engine_cost  | PRIMARY    |         2 |
 | mysql        | proxies_priv | PRIMARY    |         1 |
 | mysql        | server_cost  | PRIMARY    |         6 |
 | mysql        | tables_priv  | PRIMARY    |         2 |
 | mysql        | user         | PRIMARY    |         7 |
 | test         | sbtest1      | PRIMARY    |      2500 |
 | test         | sbtest10     | PRIMARY    |      3007 |
 | test         | sbtest100    | PRIMARY    |      2642 |
 | test         | sbtest11     | PRIMARY    |      2091 |
 +--------------+--------------+------------+-----------+
```

The following table describes parameters of the responses to the sample query requests.

Parameter	Description
TABLE_SCHEMA	The name of a database.

Parameter	Description
TABLE_NAME	The name of a table.
ROWS_READ	The number of rows read from a table.
ROWS_CHANGED	The number of rows modified in a table.
ROWS_CHANGED_X_INDEXES	The number of rows modified by using indexes in a table.
ROWS_INSERTED	The number of rows inserted into a table.
ROWS_DELETED	The number of rows deleted from a table.
ROWS_UPDATED	The number of rows updated in a table.
INDEX_NAME	The name of an index.

Use Performance Point

1. View the global variable settings of your RDS instance, as shown in the following example:

```
mysql> show variables like "%performance_point%";
 +--------------------------------------+-------+
 | Variable_name                        | Value |
 +--------------------------------------+-------+
 | performance_point_dbug_enabled       | OFF   |
 | performance_point_enabled            | ON    |
 | performance_point_iostat_interval    | 2     |
 | performance_point_iostat_volume_size | 10000 |
 | performance_point_lock_rwlock_enabled | ON   |
 +--------------------------------------+-------+
```

Note If these variables cannot be found, check that your RDS instance is running MySQL 5.7.

2. Query the events_statements_summary_by_digest_supplement table in the performance_schema database to obtain the top 10 SQL statements in various dimensions. Example:

```
mysql> select * from events_statements_summary_by_digest_supplement limit 10;
 +--------------------+----------------------------------+-------------------------------------------+------------
--+
 | SCHEMA_NAME        | DIGEST                           | DIGEST_TEXT                               | ELAPSED_TIME | ......
 +--------------------+----------------------------------+-------------------------------------------+------------
--+
 | NULL               | 6b787dd1f9c6f6c5033120760a1a82de | SELECT @@`version_comment` LIMIT ?        |
932 |
 | NULL               | 2fb4341654df6995113d998c52e5abc9 | SHOW SCHEMAS                              |       2363
|
 | NULL               | 8a93e76a7846384621567fb4daa1bf95 | SHOW VARIABLES LIKE ?                     |        17
933 |
 | NULL               | dd148234ac7a20cb5aee7720fb44b7ea | SELECT SCHEMA ( )                         |       1006
|
 | information_schema | 2fb4341654df6995113d998c52e5abc9 | SHOW SCHEMAS                              |
2156 |
 | information_schema | 74af182f3a2bd265678d3dadb53e08da | SHOW TABLES                               |
3161 |
 | information_schema | d3a66515192fcb100aaef6f8b6e45603 | SELECT * FROM `TABLE_STATISTICS`
LIMIT ? |       2081 |
 | information_schema | b3726b7c4c4db4b309de2dbc45ff52af | SELECT * FROM `INDEX_STATISTICS`
LIMIT ? |       2384 |
 | information_schema | dd148234ac7a20cb5aee7720fb44b7ea | SELECT SCHEMA ( )                         |
129 |
 | test               | 2fb4341654df6995113d998c52e5abc9 | SHOW SCHEMAS                              |        342 |
 +--------------------+----------------------------------+-------------------------------------------+------------
--+
```

The following table describes parameters of the response to the sample query request.

Parameter	Description
SCHEMA_NAME	The name of a database.
DIGEST	The 64-byte hash string obtained from the **DIGEST_TEXT** parameter.
DIGEST_TEXT	The digest of an SQL statement.
ELAPSED_TIME	The total time spent executing an SQL statement. Unit: μs.
CPU_TIME	The time spent by CPU executing an SQL statement. Unit: μs.
SERVER_LOCK_TIME	The time occupied by metadata locks on the server during the execution of an SQL statement. Unit: μs.

Parameter	Description
TRANSACTION_LOCK_TIME	The time occupied by storage transaction locks during the execution of an SQL statement. Unit: μs.
MUTEX_SPINS	The number of mutex spins triggered during the execution of an SQL statement.
MUTEX_WAITS	The number of spin waits triggered by mutexes during the execution of an SQL statement.
RWLOCK_SPIN_WAITS	The number of spin waits triggered by readers-write locks during the execution of an SQL statement.
RWLOCK_SPIN_ROUNDS	The number of rounds in which the background thread looped in the spin-wait cycles triggered by readers-write locks during the execution of an SQL statement.
RWLOCK_OS_WAITS	The number of operating system waits triggered by readers-write locks during the execution of an SQL statement.
DATA_READS	The number of times the system read data from data files during the execution of an SQL statement.
DATA_READ_TIME	The time spent reading data from data files during the execution of an SQL statement. Unit: μs.
DATA_WRITES	The number of times the system wrote data into data files during the execution of an SQL statement.
DATA_WRITE_TIME	The time spent writing data into data files during the execution of an SQL statement. Unit: μs.
REDO_WRITES	The number of times the system wrote data into log files during the execution of an SQL statement.
REDO_WRITE_TIME	The time spent writing data into log files during the execution of an SQL statement. Unit: μs.
LOGICAL_READS	The number of times the system read logical pages during the execution of an SQL statement.
PHYSICAL_READS	The number of times the system read physical pages during the execution of an SQL statement.
PHYSICAL_ASYNC_READS	The number of times system read physical asynchronous pages during the execution of an SQL statement.

3. Query the IO_STATISTICS table in the information_schema database to obtain information about recent data read and write operations: Example:

```
mysql> select * from IO_STATISTICS limit 10;
 +---------------------+-----------+----------------+
 | TIME                | DATA_READ | DATA_READ_TIME | ......
 +---------------------+-----------+----------------+
 | 2019-08-08 09:56:53 |        73 |            983 |
 | 2019-08-08 09:56:57 |         0 |              0 |
 | 2019-08-08 09:59:17 |         0 |              0 |
 | 2019-08-08 10:00:55 |      4072 |          40628 |
 | 2019-08-08 10:00:59 |         0 |              0 |
 | 2019-08-08 10:01:09 |       562 |           5800 |
 | 2019-08-08 10:01:11 |       606 |           6910 |
 | 2019-08-08 10:01:13 |       609 |           6875 |
 | 2019-08-08 10:01:15 |       625 |           7077 |
 | 2019-08-08 10:01:17 |       616 |           5800 |
 +---------------------+-----------+----------------+
```

The following table describes parameters of the response to the query request.

Parameter	Description
TIME	The point in time at which data read and write operations were performed.
DATA_READ	The number of times the system read data.
DATA_READ_TIME	The total time spent reading data. Unit: μs.
DATA_READ_MAX_TIME	The maximum time spent reading data. Unit: μs.
DATA_READ_BYTES	The total amount of data read. Unit: bytes.
DATA_WRITE	The number of times the system wrote data.
DATA_WRITE_TIME	The total time spent writing data. Unit: μs.
DATA_WRITE_MAX_TIME	The maximum time spent writing data. Unit: μs.
DATA_WRITE_BYTES	The total amount of data written. Unit: bytes.

5.Security

5.1. Data Protect

This topic describes the data protection feature provided by ApsaraDB RDS for MySQL. This feature controls permissions to perform high-risk operations.

Prerequisites

The instance runs one of the following MySQL versions:

- MySQL 8.0 (The kernel version is 20200430 or later.)
- MySQL 5.7 (The kernel version is 20200430 or later.)
- MySQL 5.6 (The kernel version is 20200430 or later.)

Context

Data protection takes effect on the following database operation commands:

- High-risk data operation commands
  - `Drop Table`
  - `Truncate Table`
  - `Alter Table Drop Paritition`
  - `Alter Table Truncate Partition`
  - `Alter Table Exchange Paritition`
  - `Drop Tablespace`
- Extended commands
  - `DROP View`
  - `ALTER View`
  - `Drop Function`
  - `Drop Procedure`
  - `Drop Trigger`
  - `Purge Binary Logs`

> **Note** Data protection is applied to the extended commands to ensure the running of application code.

Parameters

The data protection feature involves the following four parameters:

- rds_data_protect_level

  Specifies the level of data protection. Valid values:

  - NONE: disables data protection.
  - DDL: blocks DROP and TRUNCATE operations on databases and tables.
  - ALL: blocks all DROP and TRUNCATE operations, including the operations on views, stored procedures, functions, and triggers.

  > **Note** We recommend that you configure a data protection level in the non-maintenance or non-publishing phase and disable data protection in the maintenance or publishing phase.

- rds_data_protect_ignore

  Specifies a list of databases that do not need to be protected. For example, this parameter can be used in scenarios where development and production databases are created on the same RDS instance. You can specify that the development databases are not protected.

- rds_data_protect_admin

  Specifies which users can delete data when the rds_data_protect_control parameter is set to USER.

- rds_data_protect_control

  Specifies a data protection policy. The following protection policies are supported:

  - USER: Only users specified by the rds_data_protect_admin parameter or users who have the SUPER_ACL permissions can delete data. This value applies to most business scenarios on the cloud.
  - SUPER: Only users who have the SUPER_ACL permissions can delete data. You can use SUPER_ACL to implement precise data protection for common on-premises applications.
  - MAINTAIN: Only users with the SUPER_ACL and MAINTAIN permissions can delete data. The MAINTAIN permissions allow users to initiate connections from Alibaba Cloud. This value applies to scenarios where you want to delete data on Alibaba Cloud.
  - LOCAL: Only users with the SUPER_ACL and MAINTAIN permissions can delete data by logging on to the instance over local connections. This value applies to core applications. If you configure this value, you cannot delete data by logging on to the instance over remote connections. You must log on to the physical server.

Enable data protection

Data protection is in the invitational preview. You can to enable this feature.

5.2. Recycle bin

Data definition language (DDL) statements cannot be rolled back. If a table is unintentionally deleted by using a DROP TABLE statement, the table data may be lost. Alibaba Cloud provides the recycle bin feature that allows you to temporarily store deleted tables. You can specify a retention period within which you can retrieve the deleted tables. In addition, Alibaba Cloud provides the DBMS_RECYCLE package that is used to manage the deleted tables in the recycle bin.

Parameters

The following table describes the parameters that you must configure for the recycle bin feature.

Parameter	Description
loose_recycle_bin	Specifies whether to enable the recycle bin feature. You can enable this feature for your RDS instance or a specific session. You can reconfigure this parameter in the ApsaraDB for RDS console.
loose_recycle_bin_retention	The period for which you want to retain tables in the recycle bin. Unit: seconds. Default value: 604800. The default value indicates seven days. You can reconfigure this parameter in the ApsaraDB for RDS console.
recycle_scheduler	Specifies whether to enable the thread that is used to asynchronously delete tables from the recycle bin. This parameter is temporarily unavailable.
recycle_scheduler_interval	The polling interval that is followed by the thread to asynchronously delete tables from the recycle bin. Unit: seconds. Default value: 30. This parameter is temporarily unavailable.
recycle_scheduler_purge_table_print	Specifies whether to log the operations that are performed by the thread to asynchronously delete tables from the recycle bin. This parameter is temporarily unavailable.

Introduction

- Recycling and deletion
  - Recycling

    When you execute a `TRUNCATE TABLE` statement to delete a table, the system moves the deleted table to the recycle bin. Then, the system creates an empty table that has the same structure as the deleted table. The empty table resides in the same location as the deleted table.

    When you execute a `DROP TABLE or DROP DATABASE` statement to delete a table or a database, the system moves only the deleted tables to the recycle bin. The system deletes the other objects based on the following policies:

    - If no relationships exist between an object and the deleted tables, the system determines whether to retain the object based on the executed statement.
    - If an object is based on the deleted tables and may cause modifications to the data in these tables, the system deletes the object. These objects include triggers and foreign keys. The system does not delete column statistics. These statistics are stored to the recycle bin with the deleted tables.
  - Deletion

    The recycle bin starts a background thread to asynchronously delete tables from the recycle bin. These tables are stored in the recycle bin longer than the period that is specified by the **recycle_bin_retention** parameter. If a table in the recycle bin is large, the system starts another background thread to asynchronously delete the large table.
- Permission control

  When you start your RDS instance that runs MySQL, a database named __recycle_bin__ is initialized to store the data that is moved to the recycle bin. The __recycle_bin__ database is a system database. You cannot modify or delete the database.

You cannot delete tables from the recycle bin by executing `DROP TABLE` statements. However, you can use the `call dbms_recycle.purge_table('<TABLE>');` method to delete tables from the recycle bin.

> **Note** The account that you use must have the permissions to delete tables from your RDS instance and the recycle bin by executing DROP TABLE statements.

- Table naming in the recycle bin

  Tables in the __recycle_bin__ database originate from different databases and may have the same name. To ensure that each table has a unique name in the recycle bin, Alibaba Cloud implements the following naming conventions:

  ```
  "__" + <Storage Engine> + <SE private id>
  ```

  The following table describes the parameters in the naming conventions.

Parameter	Description
Storage Engine	The name of the storage engine that is used by the table.
SE private id	The unique value that is generated by the storage engine to identify the table. For example, the unique value that is used to identify an InnoDB table is the ID of the table.

- Independent recycling

  The recycle bin configuration that you specify on an RDS instance is applied only to that instance. Therefore, the recycle bin configuration that you specify on your primary RDS instance will not be applied to its secondary, read-only, or disaster recovery RDS instances to which binary logs are replicated. For example, you can specify a 7-day retention period on your primary RDS instance and a 14-day retention period on the secondary RDS instances separately.

  > **Note** The storage usage of an RDS instance varies based on the retention period that you specify on that instance.

Precautions

- After you execute a `DROP TABLE` statement to delete a table, the system may migrate the related data file from the tablespace that stores the table. This applies if the __recycle_bin__ database and the table reside in different file systems. In addition, this process is time-consuming.
- A general tablespace may store more than one table. If you execute a DROP TABLE statement to delete a table from a general tablespace, the system does not migrate the related data file from the general tablespace.

Prerequisites

Your RDS instance runs MySQL 8.0.

Manage the recycle bin

AliSQL provides the following two management methods in the DBMS_RECYCLE package:

- show_tables

  Displays all of the tables that are temporarily stored in the recycle bin. The following code snippet is an example of the show_tables method:

  ```
  call dbms_recycle.show_tables();
  ```

  Example:

  ```
  mysql> call dbms_recycle.show_tables();
  +-----------------+---------------+---------------+--------------+---------------------+---------------------+
  | SCHEMA          | TABLE         | ORIGIN_SCHEMA | ORIGIN_TABLE | RECYCLED_TIME       | PURGE_TIME          |
  +-----------------+---------------+---------------+--------------+---------------------+---------------------+
  | __recycle_bin__ | __innodb_1063 | product_db    | t1           | 2019-08-08 11:01:46 | 2019-08-15 11:01:46 |
  | __recycle_bin__ | __innodb_1064 | product_db    | t2           | 2019-08-08 11:01:46 | 2019-08-15 11:01:46 |
  | __recycle_bin__ | __innodb_1065 | product_db    | parent       | 2019-08-08 11:01:46 | 2019-08-15 11:01:4
  6 |
  | __recycle_bin__ | __innodb_1066 | product_db    | child        | 2019-08-08 11:01:46 | 2019-08-15 11:01:46
  |
  +-----------------+---------------+---------------+--------------+---------------------+---------------------+
  4 rows in set (0.00 sec)
  ```

Parameter	Description
SCHEMA	The name of the database that stores the table after the table is moved to the recycle bin.
TABLE	The name of the table after the table is moved to the recycle bin.
ORIGIN_SCHEMA	The name of the database that stores the table before the table is moved to the recycle bin.
ORIGIN_TABLE	The name of the table before the table is moved to the recycle bin.
RECYCLED_TIME	The time when the table was moved to the recycle bin.
PURGE_TIME	The time when the table is expected to be deleted from the recycle bin.

- purge_table

  Manually deletes a table from the recycle bin. The following code snippet is an example of the purge_table method:

  ```
  call dbms_recycle.purge_table('<TABLE>');
  ```

Note

- The TABLE variable specifies the new name of the table after the table is moved to the recycle bin.
- The account that you use must have the permissions to delete tables from your RDS instance and the recycle bin by executing DROP TABLE statements.

Example:

```
mysql> call dbms_recycle.purge_table('__innodb_1063');
Query OK, 0 rows affected (0.01 sec)
```

6.Best practices

6.1. Convert the storage engine of DRDS from InnoDB to X-Engine

This topic describes how to convert the storage engine of DRDS from InnoDB to X-Engine.

Context

Most users of existing ApsaraDB for RDS instances want to use X-Engine. These users have the following characteristics:

- Most RDS instances run MySQL 5.6 or MySQL 5.7. Few RDS instances run MySQL 8.0.
- A single instance has a large volume of data, which reaches the upper limit of disk space supported by the instance type. For example, an RDS instance with 4 CPU cores and 8 GB of memory supports up to 2 TB of local SSDs.
- The users also use DRDS. In addition, some users use an old version of DRDS and have customized functions, such as SQL passthrough.

To address the user requirements, Alibaba Cloud allows you to convert the storage engine of DRDS from InnoDB to X-Engine by following the procedure in this topic.

Note For more information about X-Engine, see X-Engine overview.

Conversion plan

RDS for MySQL 8.0 instances provide consistent API operations and user experience regardless of whether they use InnoDB or X-Engine. In this situation, after a DRDS upgrade, we recommend that you convert the storage engines for the RDS instances from InnoDB to X-Engine one by one. For example, if a DRDS instance has eight RDS instances, first convert the storage engine for one of the eight RDS instances. Monitor the instance running for a period of time. If you confirm that no compatibility or performance issues occur, convert the storage engines for the remaining seven RDS instances.

Compression ratio verification before conversion

Before conversion, we recommend that you purchase an RDS instance that is powered by X-Engine with the same specifications as the existing RDS instance that is powered by the InnoDB storage engine. Then, you can use Alibaba Cloud Data Transmission Service (DTS) to import data from the existing RDS instance to the purchased instance. This way, you can check the compression efficiency. The compression efficiency allows you to determine the following items:

- Instance storage capacity

  Based on the compression efficiency, you can determine the instance specifications that you need to purchase after you convert a storage engine from InnoDB to X-Engine. For example, if the space required after compression is below 30% of the original space, you can purchase an RDS instance with 1 TB of disk space after you convert the storage engine of an RDS instance that originally requires 3 TB of disk space. Alternatively, you can purchase an RDS instance with the same specifications to reserve storage space for future business development.

- Number of database shards

  After storage space is reduced, you can reduce the number of database shards. For example, you can merge databases that are distributed across instances to one instance. This reduces costs.

Note You can release the RDS instance that is powered by X-Engine after you complete the verification, or you can clear the instance for official conversion later.

Conversion procedure

1. Upgrade DRDS to a version later than V5.4.2-15744202.

   Note

   - If the DRDS version is later than v5.4.2-15744202, skip this step.
   - To ensure compatibility, you must adjust the service code. This applies if your business is based on specific API operations that are provided in an earlier version of DRDS, for example, the SQL passthrough function for performance optimization.

2. Select an RDS instance with the InnoDB storage engine from DRDS as the first instance for conversion. Export table creation statements and change the engine type to X-Engine. Then, create an RDS instance with the target specifications and X-Engine. Alternatively, use the RDS instance that you created when you verify the compression efficiency and import the table structure scripts into this instance.

   - For more information about how to create an RDS instance, see Create an ApsaraDB RDS for MySQL instance.
   - For more information about how to import and export table creation statements, see Convert the storage engine from InnoDB, TokuDB, or MyRocks to X-Engine.

   Note If you use DTS to migrate data, the engine type of the source instance is inherited by default. You must separately export the table creation statements and change the engine type to X-Engine before you can migrate data to the destination instance that is powered by X-Engine.

3. Use DTS to synchronize data from the RDS instance with the InnoDB storage engine to the RDS instance with X-Engine. For more information about data synchronization, see Configure two-way data synchronization between ApsaraDB RDS for MySQL instances.

   Note You can use the two-way synchronization function of DTS to ensure data consistency between the two RDS instances.

4. Modify DRDS routing rules and redirect the access requests to the RDS instance with the InnoDB storage engine to the RDS instance with X-Engine. If you want to modify the DRDS routing rules, .

> **Note** Run the first RDS instance with X-Engine for five days. Monitor the instance running and consider the request processing time, exception information, and two-way synchronization progress. If an exception occurs, you must make sure that services can be switched back to the original RDS instance with the InnoDB storage engine. For more information, see View the resource and engine metrics of an ApsaraDB RDS for MySQL instance.

5. After you confirm that the first RDS instance with X-Engine is running as normal, proceed with the conversion for 30% to 50% of the remaining RDS instances and then monitor the instance running for three to five days. In this case, repeat the preceding steps 2 to 4.

   > **Note** Do not release or bring the original RDS instances with InnoDB offline because these instances are required to perform DTS two-way synchronization with new RDS instances with X-Engine.

6. Perform the conversion for all the remaining RDS instances. After you complete the conversion for all the instances of DRDS, monitor the instance running for three to five days. If the new instances run as normal, release all DTS synchronization links and the original RDS instances with InnoDB.

6.2. DingTalk secures App Store top rank with X-Engine

This topic describes how X-Engine of ApsaraDB for RDS helps reduce the costs of DingTalk and implement online collaborative offices.

Context

DingTalk is a leading enterprise-grade instant messaging (IM) tool in China. It serves hundreds of millions of users across China. Its basic functions include video conferences and daily reports. DingTalk Open Platform also provides various office automation (OA) applications to facilitate communication between co-workers.

In 2020, COVID-19 is a serious problem. To avoid the risk of infection caused by work in centralized offices, a large number of employees have opted to work from home. The demand for collaborative office tools suddenly increases. In this situation, DingTalk is quickly elevated to the top of the App Store download list. This results in a sharp increase in DingTalk access. DingTalk is based on the elastic infrastructure provided by Alibaba Cloud. This ensures that all the traffic peaks are smoothly handled.

To serve a large number of users, DingTalk must ensure the timely and correct delivery of messages, and provide specific functions, such as read and unread messages. Unlike user-level IM tools such as WeChat, enterprise-grade IM tools must include the permanent storage of chat records and provide the multi-terminal roaming function. This function allows users to receive messages from multiple terminals. As the number of users sharply increases, DingTalk faces challenges in the costs incurred by the permanent storage of chat records while ensuring the performance of read and write operations on the chat records.

To address these challenges, DingTalk uses X-Engine as the storage engine for messages. This achieves a balance between the costs and performance. X-Engine has the following advantages:

- The storage space required by X-Engine is about 62% less than that required by the InnoDB storage engine.
- Specific database functions such as transactions and secondary indexes are supported.
- Service code can be migrated to RDS instances that are powered by X-Engine without changes.
- X-Engine separates hot and cold data to accelerate the processing of current messages. It also implements the most efficient compression algorithm for historical messages.

X-Engine storage efficiency is tested on two datasets: Link-Bench and Alibaba internal transaction business. In the test, X-Engine requires 2-fold less storage space than the InnoDB storage engine with compression enabled, and 3- to 5-fold less storage space than the InnoDB engine with compression disabled.

Low costs achieved by X-Engine

X-Engine adopts the following technologies to ensure low costs:

- Compact pages

  X-Engine uses the Copy-on-write technology to write new data to new pages without updating the original pages. The new pages are read-only and cannot be directly updated. These pages are stored in a compact manner, and the data is compressed by using specific algorithms, such as prefix encoding. This improves the storage efficiency. You can use the compaction operation to clear invalid records. This ensures a compact arrangement of valid records. X-Engine requires only 10% to 50% of the storage space compared with conventional storage engines, such as InnoDB.

- Data compression and cleaning of invalid records

  Pages after encoding can be compressed by using general compression algorithms, such as zlib, zstd, and snappy. Data at a low level of a log-structured merge-tree (LSM tree) is compressed by default.

  Data compression sacrifices computing resources for storage space. We recommend that you select compression algorithms that provide a low compression ratio and a high speed of compression and decompression. After a large number of comparative tests, X-Engine selects zstd as the default compression algorithm with additional support for other compression algorithms.

  In addition, the compaction operation is introduced to delete invalid records. This way, only valid records are retained. The more frequently the compaction operation is performed, the lower the proportion of invalid records, and the higher the storage efficiency. Therefore, you must perform the compaction operation at a suitable frequency.

  The X-Engine team also develops the field-programmable gate array (FPGA) compaction technology to reduce the computing resource consumption of the compaction operation. This technology uses heterogeneous computing hardware to accelerate the compaction process. It streamlines compaction and compression operations by using FPGA hardware. On a host without FPGA hardware, X-Engine can use a suitable scheduling algorithm to save storage space at a lower performance cost.

- Intelligent separation between hot and cold data

In normal access to a storage system, most access requests direct to a small portion of data. This is why the cache works. In an LSM tree structure, frequently accessed data is stored at a high level to a fast storage device, such as NVM and DRAM. Infrequently accessed data is stored at a low level to a slow storage device. This is the hot and cold data separation in X-Engine.

The separation algorithm completes the following tasks:

- In the compaction operation, the pages and records that are least likely to be accessed are selected and moved to the bottom of the LSM tree.
- Current hot data is selected and backfilled to memory (BlockCache and RowCache) in the compaction or dump process. This prevents compromised performance from jitters in cache hit rates.
- The AI algorithm recognizes data that may be accessed in the future and pre-reads it into memory. This increases the hit rates for accessing cache at the first time.

Hot data and cold data are accurately identified to avoid computing resource waste due to invalid compression or decompression. This improves system throughput.

For more information, see X-Engine overview.

Related papers

- X-Engine: An Optimized Storage Engine for Large-scale E-commerce Transaction Processing
- FPGA-Accelerated Compactions for LSM-based Key-Value Store

6.3. Storage engine that processes trillions of Taobao orders

Taobao historical orders are supported by a PolarDB-X cluster based on X-Engine. This fixes the known issues caused by the use of HBase databases, reduces storage costs, and allows users to query orders at all times.

Context

Taobao is the largest online shopping platform in China. It serves more than 700 million active users and tens of millions of sellers.

The large platform provides support for about 100 million transactions on physical and virtual commodities every day. Each transaction process involves different phases, such as member information verification, commodity library inquiry, order creation, inventory reduction, discounts, order payment, logistics information update, and payment confirmation. Each phase involves database record creation and status update. The entire process requires hundreds of database transactions, and the entire database cluster performs tens of billions of transaction read and write operations every day. The database team faces the challenge of huge storage costs incurred by the increasing volume of data every day while ensuring the stable performance of the database system.

Orders are the most critical information in the entire transaction process and must be permanently stored in databases. If a transaction dispute arises, the order records must be provided for users to query. Over the last nearly 17 years since Taobao was founded in 2003, the total number of database records related to orders has reached the trillion level, and the disk space occupied by these records has exceeded the PB level.

The following sections describe how Taobao ensures low latency every time that users query orders without increasing storage costs.

Architecture evolution

The architecture of transaction order databases has evolved through four phases as the traffic increases.

- Phase 1

  In this initial phase, the traffic was low, and Taobao used an Oracle database to store all order information. Order creation and historical order queries were performed on the same database.

- Phase 2

  As the volume of historical order data increased, the single database can no longer meet the performance and capacity requirements at the same time. Therefore, the database was split into an online database and a historical database. Historical orders that were generated three months ago were migrated to the historical database. However, the historical database contained too much data to allow queries. In this phase, users can only query historical orders for the last three months.

- Phase 3

  To fix the issues related to storage costs and historical order queries, Taobao migrated historical orders to an HBase database.

  HBase provides both primary and indexing tables. Users can query the primary tables for order details and the indexing tables for order IDs based on the IDs of buyers or sellers. In this situation, orders may not be migrated to the historical order database in chronological order, and many types of orders are not migrated to this database. As a result, the order list is not sorted by time, and users cannot search for orders by using the listed sequence of orders.

- Phase 4

  The historical order database is built in a PolarDB-X cluster based on X-Engine. This reduces storage costs and fixes the out-of-time-order issue.

Business pain points

The architecture evolution shows that the business team and the database team have suffered from the following pain points over the last 10 years since the historical order database was introduced:

- Storage costs

  A large volume of data is written every day and the data is never deleted. Low-cost storage is required.

- Query performance

  Diversified query functions are required to meet specific requirements, such as query by time and query by order type. Databases must support secondary indexes that can ensure data consistency and performance.

- Query latency

  The query latency must be low to ensure user experience. For example, queries on historical orders of 90 days ago are much fewer than those in the last 90 days, but these queries still require low latency.

Historical order database solution based on X-Engine

The transaction order system has been iterated for 10 years in terms of the architecture, where online and historical databases are separated. Most service code is compatible with this architecture, which is also inherited in this solution. This architecture mitigates risks caused by the reconstruction and migration of service code. Initially, the HBase cluster is replaced with the PolarDB-X cluster that is based on X-Engine.

- The online database is still deployed in a MySQL cluster that is based on the InnoDB storage engine, and stores only orders for the last 90 days. The data volume is small, which ensures a high cache hit rate and reduces read/write latency.
- Orders that were generated 90 days ago are migrated from the online database to the historical database through data synchronization and are deleted from the online database.
- The storage engine of the historical database is converted to X-Engine. This database stores all orders that were generated 90 days ago. If you want to perform read or write operations on these orders, access the historical database.

After this new solution is used, the storage costs are the same as those incurred by the use of the HBase database. The historical database is compatible with the online database, and identical indexes can be created on the two databases. This fixes the out-of-time-order issue. In the historical database, hot data is separated from cold data to reduce read latency.

Summary

The transaction order records on Taobao are stored in the streamline mode. Recently written records are frequently accessed at first, and the access frequency sharply decreases over time. X-Engine separates hot and cold data and is suitable for this type of access. A single database cluster based on X-Engine is sufficient for these access scenarios.

Assume that a new or existing business needs to store a large number of streamline records. If hot data and cold data are not separated on the business layer, we recommend that you use the distributed PolarDB-X cluster based on X-Engine to ensure scalability without increasing storage costs.

Alibaba Cloud has launched X-Engine. You can purchase it if required. For more information, see Create an ApsaraDB RDS for MySQL instance.

6.4. Best practices for X-Engine testing

This topic describes how to use SysBench to test the performance of X-Engine used with ApsaraDB RDS for MySQL. This helps you evaluate the performance of X-Engine.

Prerequisites

- The default storage engine of your RDS instance is X-Engine.

Note If X-Engine is used, the value of the XENGINE parameter must be DEFAULT in the Support column.

```
MySQL [(none)]> show storage engines;
+--------------------+---------+----------------------------------------------------------------+--------------
+------+------------+
| Engine             | Support | Comment                                                        | Transactions | XA   | Savepoi
nts |
+--------------------+---------+----------------------------------------------------------------+--------------
+------+------------+
| FEDERATED          | NO      | Federated MySQL storage engine                                 | NULL         | NULL
| NULL       |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it disappears) | NO
| NO   | NO         |
| XENGINE            | DEFAULT | X-Engine storage engine                                        | YES          | YES  | YE
S          |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables      | NO
| NO   | NO         |
| InnoDB             | YES     | Supports transactions, row-level locking, and foreign keys     | YES
| YES  | YES        |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                             | NO           | NO
| NO         |
| Sequence           | YES     | Sequence Storage Engine Helper                                 | NO           | NO   | N
O          |
| MyISAM             | YES     | MyISAM storage engine                                          | NO           | NO   | NO
|
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                          | NO           | NO   |
NO         |
| CSV                | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| ARCHIVE            | YES     | Archive storage engine                                         | NO           | NO   | NO
|
+--------------------+---------+----------------------------------------------------------------+--------------
+------+------------+
11 rows in set (0.00 sec)
```

- The table used for testing is stored in X-Engine.

Note In this example, the table used for testing is created with the ENGINE parameter set to XENGINE. If you set the ENGINE parameter to INNODB or another storage engine, the table used for testing is stored in the specified storage engine rather than X-Engine.

```
MySQL [sbtest]> show create table sbtest1;
+---------+-------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
---------------------------+
| Table   | Create Table
|
+---------+-------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
---------------------------+
| sbtest1 | CREATE TABLE `sbtest1` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `k` int(11) NOT NULL DEFAULT '0',
  `c` char(120) COLLATE utf8mb4_general_ci NOT NULL DEFAULT '',
  `pad` char(60) COLLATE utf8mb4_general_ci NOT NULL DEFAULT '',
  PRIMARY KEY (`id`),
  KEY `k_1` (`k`)
) ENGINE=XENGINE AUTO_INCREMENT=2001 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_gener
al_ci |
+---------+-------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
---------------------------+
1 row in set (0.01 sec)
```

Note We recommend that you use SysBench 1.1.0 or later.

Use DTS to test storage space usage

We recommend that you use Alibaba Cloud Data Migration Service (DTS) to migrate your actual database data to your RDS instance and then check the disk usage of X-Engine. In this case, the test results are more close to your actual business situation. X-Engine adopts technologies such as space-friendly storage format, prefix encoding, tiered storage, and efficient compression algorithms to reduce disk usage. The actual effect of these technologies varies based on schemas and record length in your databases. Therefore, using your actual database data allows you to obtain more accurate test results.

DTS does not support an automatic conversion of the storage engine during data migration. You must manually create databases and tables on your RDS instance that runs X-Engine, set the ENGINE parameter to XENGINE in the table creation statements as described in the "Prerequisites" section, and migrate data by using DTS. Do not migrate the schemas.

We recommend that you do not execute SQL statements immediately after the data import is complete. You can monitor the CPU utilization and input/output operations per second (IOPS) usage of your RDS instance in the ApsaraDB for RDS console. After the CPU utilization and IOPS approach zero, you can execute SQL statements to test the performance of X-Engine. In this case, the disk usage is calculated more accurately. This is because the log-structured merge-tree (LSM tree) architecture used by X-Engine depends on background asynchronous tasks to implement functions such as data compression. These functions reduce storage costs. The background asynchronous tasks take some time and consume CPU and IOPS resources.

For more information, see Migrate data between ApsaraDB for RDS instances.

Use SysBench to test the storage space usage

To fully test the compression efficiency of X-Engine, we recommend that you set the table_size parameter in the following command to a large value within the disk size range allowed for your RDS instance.

We recommend that you monitor the CPU utilization and IOPS usage of your RDS instance in the ApsaraDB for RDS console. After the CPU utilization and IOPS approach zero, the space usage is calculated more accurately.

Run the following command to test the storage space usage:

```
sysbench /usr/share/sysbench/oltp_update_index.lua \
 -- mysql-host=[The endpoint of your RDS instance] \
 --mysql-user=sbtest \
 --mysql-password=sbtest@888 \
 --mysql-db=sbtest \
 --threads=32 \
 --tables=32 \
 --table_size=1000000000 \
 --mysql-storage-engine=XENGINE \
 prepare
```

Use SysBench to test performance

If you use SysBench for performance testing, we recommend that you set the rand-type parameter to zipfian and the rand-zipfian-exp parameter to 0.9.

- rand-type: specifies the type of the distribution that is used to generate random numbers in SQL statements.
- Zipfian distribution: a common data distribution with hot spots. When the rand-zipfian-exp parameter is set to 0.9, the random numbers generated by using Zipfian distribution are closer to those generated in the real world. The test results are more valuable in comparison to those generated by using the default uniform distribution.

We recommend that you conduct a single test for a long period of time, such as 3,600 seconds. The test results of average performance obtained from a long-time test is more valuable and less affected by potential interference factors.

We recommend that you use a large number of threads, for example, 512 threads, to test the throughput.

To improve the performance of X-Engine by configuring parameters, contact your Alibaba Cloud account manager or after-sales engineers. You can also to receive consulting services.

Run the following command to test the performance:

```
sysbench /usr/share/sysbench/oltp_point_select.lua \
  -- mysql-host=[The endpoint of your RDS instance] \
  --mysql-user=sbtest \
  --mysql-password=sbtest@888 \
  --time=3600 \
  --mysql-db=sbtest \
  --tables=32 \
  --threads=512 \
  --table_size=10000000 \
  --rand-type=zipfian \
  --rand-zipfian-exp=0.9 \
  --report-interval=1 \
  run
```

Use a Python script to perform multiple tests at a time

If you want to perform multiple tests at a time by using SysBench, we recommend that you use a Python script that can automatically perform the tests and record the test results. When you execute the script, you are prompted to enter the endpoint of your RDS instance. Example:

```
import subprocess
import time
import sys

def execute_test(test_name, db_conn_string):
  # setup sysbench parameters
  mysql = "--mysql-host=%s" % db_conn_string
  user = "--mysql-user=sbtest"
  password = "--mysql-password=*********"
  time = "--time=3600"
  database = "--mysql-db=sbtest"
  tables = "--tables=32"
  threads = "--threads=512"
  table_size = "--table_size=1000000"
  distribution = "--rand-type=pareto --rand-pareto-h=0.9"
  # formulate the sysbench command
  cmd = 'sysbench ' + test_name + " " + mysql + " " + user+ " " + password + " " + time+ " " + database+ "
" + tables + " " + threads + " " + table_size+ " " + distribution+ " " + "--report-interval=1"+ " " +'run'
  # execute
  out = subprocess.check_output(cmd,
    stderr = subprocess.STDOUT, shell=True)
  # output sysbench outputs to a file
  output_file_name = "xengine_result_"+test_name[20:len(test_name)]
  output_file = open(output_file_name, "w")
  output_file.write(out)
  output_file.close()

if __name__ == '__main__':
  # the connection string for the MySQL (X-Engine) instance to be tested
  db_conn_string = sys.argv[1]

  test = [
    "/usr/share/sysbench/oltp_update_index.lua",
    "/usr/share/sysbench/oltp_point_select.lua",
    "/usr/share/sysbench/oltp_read_only.lua",
    "/usr/share/sysbench/oltp_write_only.lua",
    "/usr/share/sysbench/oltp_read_write.lua",
    "/usr/share/sysbench/oltp_insert.lua"
  ]

  for atest in test:
    print("start test:\t%s\t%s" % (atest, time.ctime()))
    execute_test(atest, db_conn_string)
    print("end test:\t%s\t%s" % (atest, time.ctime()))
    # sleep foe some seconds
    # after a period of testing with inserts/updates/deletes, x-engine needs some time to complete
    # its asynchronous background compactions.
    time.sleep(1000)
```