# TiDB Release Notes: Changes from v7.6.0 to v8.1.0

PingCAP

20240524

**Abstract**

This document contains the release notes for TiDB v7.6.0-DMR, v8.0.0-DMR, and v8.1.0-LTS. When you upgrade from v7.5.x to v8.1.0, you can refer to this document for a thorough overview of new features, compatibility changes, improvements, and bug fixes.

For detailed guidance and additional resources about TiDB v8.1.0, see TiDB v8.1 documentation.

## Table of Contents

# 1 TiDB 8.1.0 Release Notes

Release date: May 24, 2024

TiDB version: 8.1.0

Quick access: Quick start | Production deployment

TiDB 8.1.0 is a Long-Term Support Release (LTS).

Compared with the previous LTS 7.5.0, 8.1.0 includes new features, improvements, and bug fixes released in 7.6.0-DMR and 8.0.0-DMR. When you upgrade from 7.5.x to 8.1.0, you can download the TiDB Release Notes PDF to view all release notes between the two LTS versions. The following table lists some highlights from 7.6.0 to 8.1.0:

| Category | Feature/Enhancement | Description |
|---|---|---|
| Scalability and Performance | Acceleration of cluster snapshot restore speed (GA in v8.0.0) | With this feature, BR can fully leverage the scale advantage of a cluster, enabling all TiKV nodes in the cluster to participate in the preparation step of data restores. This feature can significantly improve the restore speed of large datasets in large-scale clusters. Real-world tests show that this feature can saturate the download bandwidth, with the download speed improving by 8 to 10 times, and the end-to-end restore speed improving by approximately 1.5 to 3 times. |
| | Achieve up to 10 times faster for creating tables in batch (experimental, introduced in v7.6.0) | With the implementation of the new DDL architecture in v7.6.0, the performance of batch table creation has witnessed a remarkable improvement, up to 10 times faster. This substantial enhancement drastically reduces the time needed for creating numerous tables. This acceleration is particularly noteworthy in SaaS scenarios, where the prevalence of high volumes of tables, ranging from tens to hundreds of thousands, is a common challenge. |
| | Use Active PD Followers to enhance PD's Region information query | TiDB v7.6.0 introduces an experimental feature "Active PD Follower", which allows PD followers to provide Region information query services. This feature improves the capability of the PD cluster to handle `GetRegion` and `ScanRegions` requests in |

| Category | Feature/Enhancement | Description |
|---|---|---|
| | service (experimental, introduced in v7.6.0) | clusters with a large number of TiDB nodes and Regions, thereby reducing the CPU pressure on PD leaders. |
| | Bulk DML for much larger transactions (experimental, introduced in v8.0.0) | Large batch DML jobs, such as extensive cleanup jobs, joins, or aggregations, can consume a significant amount of memory and have previously been limited at very large scales. Bulk DML (tidb_dml_type = "bulk") is a new DML type for handling large batch DML tasks more efficiently while providing transaction guarantees and mitigating OOM issues. This feature differs from import, load, and restore operations when used for data loading. |
| | Enhance the stability of caching the schema information when there is a massive number of tables (experimental, introduced in v8.0.0) | SaaS companies using TiDB as the system of record for their multi-tenant applications often need to store a substantial number of tables. In previous versions, handling table counts in the order of a million or more was feasible, but it had the potential to degrade the overall user experience. TiDB v8.0.0 improves the situation with the following enhancements:<br>• Introduce a new schema information caching system, which provides a cache strategy based on the Least Recently Used (LRU) algorithm for table metadata. It prioritizes storing the metadata of frequently accessed tables in the cache, thereby reducing memory usage in scenarios with a large number of tables.<br>• Implement a priority queue for auto analyze, making the process less rigid and enhancing stability across a wider array of tables. |
| Reliability and availability | Global Sort (GA in v8.0.0) | The Global Sort feature aims to improve the stability and efficiency of IMPORT INTO and CREATE INDEX. By globally sorting the data to be processed, this feature improves the stability, controllability, and scalability of data writing to TiKV, consequently enhancing the user experience and service quality of data import and index creation. With global |

| Category | Feature/Enhancement | Description |
|---|---|---|
| | | sorting enabled, each IMPORT INTO or CREATE INDEX statement now supports importing or adding indexes for up to 40 TiB of data. |
| | Cross-database SQL binding (introduced in v7.6.0) | When managing hundreds of databases with the same schema, it is often necessary to apply SQL bindings across these databases. For example, in SaaS or PaaS data platforms, each user typically operates separate databases with the same schema and runs similar SQL queries on them. In this case, it is impractical to bind SQL for each database one by one. TiDB v7.6.0 introduces cross-database SQL bindings that enable matching bindings across all schema-equivalent databases. |
| | Support TiProxy (GA in v8.0.0) | Full support for the TiProxy service, easily deployable via deployment tooling, to manage and maintain connections to TiDB so that they live through rolling restarts, upgrades, or scaling events. |
| | Data Migration (DM) officially supports MySQL 8.0 (GA in v7.6.0) | Previously, using DM to migrate data from MySQL 8.0 is an experimental feature and is not available for production environments. TiDB v7.6.0 enhances the stability and compatibility of this feature to help you smoothly and quickly migrate data from MySQL 8.0 to TiDB in production environments. In v7.6.0, this feature becomes generally available (GA). |
| | TiDB resource control supports managing queries that consume more resources than expected (GA in v8.1.0) | Through the rules of resource groups, TiDB can automatically identify queries that consume more resources than expected, and then limit or cancel these queries. Even if the queries are not identified by the rules, you can still manually add query characteristics and take corresponding measures to reduce the impact of the sudden query performance problem on the entire database. |
| DB Operations and Observability | Support monitoring index usage statistics (introduced in v8.0.0) | Proper index design is a crucial prerequisite to maintaining database performance. TiDB v8.0.0 introduces the INFORMATION_SCHEMA.TIDB_INDEX_USAGE table and the |

| Category | Feature/Enhancement | Description |
|---|---|---|
| | | sys.schema_unused_indexes view to provide usage statistics of indexes. This feature helps you assess the efficiency of indexes in the database and optimize the index design. |
| Data Migration | TiCDC supports the Simple protocol (introduced in v8.0.0) | TiCDC introduces a new protocol, the Simple protocol. This protocol provides in-band schema tracking capabilities by embedding table schema information in DDL and BOOTSTRAP events. |
| | TiCDC supports the Debezium format protocol (introduced in v8.0.0) | TiCDC introduces a new protocol, the Debezium protocol. TiCDC can now publish data change events to a Kafka sink using a protocol that generates Debezium style messages. |
| | TiCDC supports client authentication (introduced in v8.1.0) | TiCDC supports client authentication using mutual Transport Layer Security (mTLS) or TiDB username and password. This feature enables CLI or OpenAPI clients to authenticate their connections to TiCDC. |

## 1.1 Feature details

### 1.1.1 Reliability

- Support managing queries that consume more resources than expected (GA) #43691 @nolouch

  Sudden SQL query performance problems can cause a decline in overall database performance, which is the most common challenge to database stability. The reasons for these problems are diverse, such as untested new SQL statements, drastic changes in data volume, and sudden changes in execution plans. These problems are difficult to avoid completely at the source. TiDB v7.2.0 has introduced the capability to manage queries that consume more resources than expected to quickly reduce the impact of sudden query performance problems. This feature becomes generally available in v8.1.0.

  You can set the maximum execution time for a query in a resource group. When the execution time of a query exceeds the set value, the priority of the query is automatically reduced or the query is canceled. You can also set immediately matching identified queries through text or execution plans within a period of time, to avoid excessive resource consumption during the identification phase when the concurrency of problematic queries is too high.

TiDB also supports the manual marking of queries. By using the QUERY WATCH command, you can mark queries based on the SQL text, SQL Digest, or execution plan. The queries that match the mark can be downgraded or canceled, achieving the purpose of adding a SQL blocklist.

The automatic management capability of queries that consume more resources than expected provides users with an effective means to quickly mitigate the impact of query problems on overall performance before the root cause is identified, thereby improving the stability of the database.

For more information, see documentation.

### 1.1.2 SQL

- Support using more expressions to set default column values when creating a table (GA) #50936 @zimulala

  Before v8.0.0, when you create a table, the default value of a column is limited to strings, numbers, dates, and certain expressions. Starting from v8.0.0, you can use more expressions as the default column values. For example, you can set the default value of a column to DATE_FORMAT. This feature helps you meet more diverse requirements. In v8.1.0, this feature becomes GA.

  Starting from v8.1.0, you can use expressions as default values when adding columns by ADD COLUMN.

  For more information, see documentation.

### 1.1.3 DB operations

- Enable the TiDB Distributed eXecution Framework (DXF) by default to enhance the performance and stability of ADD INDEX or IMPORT INTO tasks in parallel execution #52441 @D3Hunter

  The DXF becomes generally available (GA) in v7.5.0, but it is disabled by default. This means that an ADD INDEX or IMPORT INTO task is executed only by one TiDB node by default.

Starting from v8.1.0, TiDB enables this feature by default (tidb_enable_dist_task defaults to ON). When enabled, the DXF can schedule multiple TiDB nodes to execute the same ADD INDEX or IMPORT INTO task in parallel, fully utilizing the resources of the TiDB cluster and greatly improving the performance of these tasks. In addition, you can linearly improve the performance of ADD INDEX and IMPORT INTO tasks by adding TiDB nodes and configuring tidb_service_scope for the newly added nodes.

For more information, see documentation.

### 1.1.4 Security

- Enhance TiDB log desensitization (GA) #52364 @xhebox

  The enhanced TiDB log desensitization supports removing sensitive data when users view logs, implemented by marking SQL text information in log files. You can control whether to mark log information to enable secure use of TiDB logs in different scenarios, enhancing the security and flexibility of using log desensitization. To use this feature, set the system variable tidb_redact_log to MARKER, and then the SQL text in TiDB's runtime logs is marked. In addition, you can use the collect-log subcommand on the TiDB server to remove marked sensitive data from the logs and display the logs in a secure manner. You can also remove all markers and get the normal logs. This feature became generally available in v8.1.0.

  For more information, see documentation.

### 1.1.5 Data migration

- Support the IMPORT INTO ... FROM SELECT syntax (GA) #49883 @D3Hunter

  Before v8.0.0, importing query results into a target table could only be done using the INSERT INTO ... SELECT statement, which is relatively inefficient in some large dataset scenarios. In v8.0.0, TiDB introduces IMPORT INTO ... FROM SELECT as an experimental feature, which enables you to import the results of a SELECT query into an empty TiDB target table. It achieves up to 8 times the performance of INSERT INTO ... SELECT and significantly reduces the import time. In addition, you can use IMPORT INTO ... FROM SELECT to import historical data queried with AS OF TIMESTAMP.

In v8.1.0, the IMPORT INTO ... FROM SELECT syntax becomes generally available (GA), enriching the functionality scenarios of the IMPORT INTO statement.

For more information, see documentation.

- TiDB Lightning simplifies conflict resolution strategies and supports handling conflicting data using the replace strategy (GA) #51036 @lyzx2001

  Before v8.0.0, TiDB Lightning has one data conflict resolution strategy for the logical import mode and two data conflict resolution strategies for the physical import mode, which are not easy to understand and configure.

  In v8.0.0, TiDB Lightning deprecates the old version of conflict detection strategy for the physical import mode, enables you to control the conflict detection strategy for both logical and physical import modes via the conflict.strategy parameter (experimental), and simplifies the configuration of this parameter. In addition, in the physical import mode, the replace strategy supports retaining the latest data and overwriting the old data when the import detects data with primary key or unique key conflicts. In v8.1.0, the capability to handle conflicting data with the replace strategy becomes generally available (GA).

  For more information, see documentation.

- TiCDC supports client authentication #10636 @CharlesCheung96

  In v8.1.0, TiCDC supports client authentication when you are using the TiCDC CLI or OpenAPI. This feature enables you to configure TiCDC to require client authentication using client certificates, thereby establishing mutual Transport Layer Security (mTLS). Additionally, you can configure authentication based on TiDB username and password.

  For more information, see documentation.

## 1.2 Compatibility changes

**Note:**

This section provides compatibility changes you need to know when you upgrade from v8.0.0 to the current version (v8.1.0). If you are upgrading from v7.6.0 or earlier versions to the current version, you might also need to check the compatibility changes introduced in intermediate versions.

### 1.2.1 Behavior changes

- In earlier versions, the tidb.tls configuration item in TiDB Lightning treats values "false" and "" the same, as well as treating the values "preferred" and "skip-verify" the same. Starting from v8.1.0, TiDB Lightning distinguishes the behavior of "false", "", "skip-verify", and "preferred" for tidb.tls. For more information, see TiDB Lightning configuration.
- For tables with AUTO_ID_CACHE=1, TiDB supports a centralized auto-increment ID allocating service. In earlier versions, the primary TiDB node of this service automatically performs a forceRebase operation when the TiDB process exits (for example, during the TiDB node restart) to keep auto-assigned IDs as consecutive as possible. However, when there are too many tables with AUTO_ID_CACHE=1, executing forceRebase becomes very time-consuming, preventing TiDB from restarting promptly and even blocking data writes, thus affecting system availability. To resolve this issue, starting from v8.1.0, TiDB removes the forceRebase behavior, but this change will cause some auto-assigned IDs to be non-consecutive during the failover.

### 1.2.2 System variables

| Variable name | Change type | Description |
|---|---|---|
| tidb_auto_analyze_ratio | Modified | Changes the value range from [0, 18446744073709551615] to (0, 1]. |
| tidb_enable_dist_task | Modified | Changes the default value from OFF to ON. This means that Distributed eXecution Framework (DXF) is enabled by default, which fully utilizes the resources of the TiDB cluster and greatly improves the performance of ADD INDEX and IMPORT INTO tasks. If you want to upgrade a cluster with the DXF enabled to v8.1.0 or later, disable the DXF (by setting tidb_enable_dist_task to OFF) before the upgrade, which avoids ADD INDEX operations during the upgrade causing data index inconsistency. After the upgrade, you can manually enable the DXF. |
| tidb_service_scope | Modified | Changes the optional value from "" or background to a string with a length of up to 64 characters, which enables you to control the service scope of each TiDB |

| Variable name | Change type | Description |
|---|---|---|
| | | node more flexibly. Valid characters include digits 0-9, letters a-zA-Z, underscores _, and hyphens -. The Distributed eXecution Framework (DXF) determines which TiDB nodes can be scheduled to execute distributed tasks based on the value of this variable. For specific rules, see Task scheduling. |

### 1.2.3 Configuration file parameters

| Configuration file | Configuration parameter | Change type | Description |
|---|---|---|---|
| TiDB | concurrently-init-stats | Newly added | Controls whether to initialize statistics concurrently during TiDB startup. The default value is false. |
| TiDB Lightning | conflict.max-record-rows | Modified | Starting from v8.1.0, there is no need to configure conflict.max-record-rows manually, because TiDB Lightning automatically assigns the value of conflict.max-record-rows with the value of conflict.threshold, regardless of the user input. conflict.max-record-rows will be deprecated in a future release. |
| TiDB Lightning | conflict.threshold | Modified | Changes the default value from 9223372036854775807 to 10000 to quickly interrupt abnormal tasks so that you can make corresponding adjustments as soon as possible. This saves time and computational resources by avoiding the scenario where a large amount of conflicting data is discovered after the import, caused by abnormal data sources or incorrect table schema definitions. |
| TiCDC | security.client-allowed-user | Newly added | Lists the usernames that are allowed for client authentication. Authentication requests with usernames not in this list will be rejected. The default value is null. |
| TiCDC | security.client-user-required | Newly added | Controls whether to use username and password for client authentication. The default value is false. |

| Configuration file | Configuration parameter | Change type | Description |
|---|---|---|---|
| TiCDC | security.mtls | Newly added | Controls whether to enable the TLS client authentication. The default value is false. |
| TiCDC | sink.debezium.output-old-value | Newly added | Controls whether to output the value before the row data changes. The default value is true. When it is disabled, the UPDATE event does not output the “before” field. |
| TiCDC | sink.open.output-old-value | Newly added | Controls whether to output the value before the row data changes. The default value is true. When it is disabled, the UPDATE event does not output the “p” field. |

## 1.3 Deprecated features

- It is planned to redesign the auto-evolution of execution plan bindings in subsequent releases, and the related variables and behavior will change.
- The TiDB Lightning parameter conflict.max-record-rows is scheduled for deprecation in a future release and will be subsequently removed. This parameter will be replaced by conflict.threshold, which means that the maximum number of conflicting records is consistent with the maximum number of conflicting records that can be tolerated in a single import task.
- Starting from v8.0.0, TiDB Lightning deprecates the old version of conflict detection strategy for the physical import mode, and enables you to control the conflict detection strategy for both logical and physical import modes via the conflict.strategy parameter. The duplicate-resolution parameter for the old version of conflict detection will be removed in a future release.

## 1.4 Improvements

- TiDB

  - Improve the MySQL compatibility of foreign keys displayed in the output of SHOW CREATE TABLE #51837 @negachov
  - Improve the MySQL compatibility of expression default values displayed in the output of SHOW CREATE TABLE #52939 @CbcWestwolf

  - Support adding multiple indexes concurrently in the ingest mode #52596 @lance6716
  - Support configuring the system variable tidb_service_scope with various values, enhancing the utilization of the Distributed eXecution Framework (DXF) #52441 @ywqzzy
  - Enhance the handling of DNF items that are always false by directly ignoring such filter conditions, thus avoiding unnecessary full table scans #40997 @hi-rustin
  - Support using Optimizer Fix Controls to remove the limitation that the optimizer does not automatically choose Index Merge for a query when the optimizer can choose the single index scan method (other than full table scan) for the query #52869 @time-and-fate
  - Add the total_kv_read_wall_time metric to the column execution info of Coprocessor operators #28937 @cfzjywxk
  - Add the RU (max) metric on the Resource Control dashboard #49318 @nolouch
  - Add a timeout mechanism for LDAP authentication to avoid the issue of resource lock (RLock) not being released in time #51883 @YangKeao
- TiKV
  - Avoid performing IO operations on snapshot files in Raftstore threads to improve TiKV stability #16564 @Connor1996
  - Accelerate the shutdown speed of TiKV #16680 @LykxSassinator
  - Add metrics for memory usage per thread #15927 @Connor1996
- PD
  - Optimize the logic for OperatorController to reduce the overhead of competition locks #7897 @nolouch
- TiFlash
  - Mitigate the issue that TiFlash might panic due to updating certificates after TLS is enabled #8535 @windtalker
- Tools
  - Backup & Restore (BR)

        - Add PITR integration test cases to cover compatibility testing for log backup and adding index acceleration #51987 @Leavrth
        - Remove the invalid verification for active DDL jobs when log backup starts #52733 @Leavrth
        - Add test cases to test compatibility between PITR and the acceleration of adding indexes feature #51988 @Leavrth
        - BR cleans up empty SST files during data recovery #16005 @Leavrth
    - TiCDC
        - Improve memory stability during data recovery using redo logs to reduce the probability of OOM #10900 @CharlesCheung96
        - Significantly improve the stability of data replication in transaction conflict scenarios, with up to 10 times performance improvement #10896 @CharlesCheung96

## 1.5 Bug fixes

- TiDB
    - Fix the issue that executing SQL statements containing tables with multi-valued indexes might return the Can't find a proper physical plan for this query error #49438 @qw4990
    - Fix the issue that automatic statistics collection gets stuck after an OOM error occurs #51993 @hi-rustin
    - Fix the issue that after using BR to restore a table that has no statistics, the statistics health of that table is still 100% #29769 @winoros
    - Fix the issue that TiDB creates statistics for system tables during upgrade #52040 @hi-rustin
    - Fix the issue that automatic statistics collection is triggered before the initialization of statistics finishes #52346 @hi-rustin
    - Fix the issue that TiDB might crash when tidb_mem_quota_analyze is enabled and the memory used by updating statistics exceeds the limit #52601 @hawkingrei
    - Fix the issue that the TiDB synchronously loading statistics mechanism retries to load empty statistics indefinitely and

prints the fail to get stats version for this histogram log #52657 @hawkingrei

- Fix the issue that expressions containing different collations might cause the query to panic when the new framework for collations is disabled #52772 @wjhuang2016
- Fix the issue that the CPS by type metric displays incorrect values #52605 @nolouch
- Fix the issue that the nil pointer error occurs when you query INFORMATION_SCHEMA.TIKV_REGION_STATUS #52013 @JmPotato
- Fix the incorrect error message displayed when an invalid default value is specified for a column #51592 @danqixu
- Fix the issue that adding indexes in the ingest mode might cause inconsistent data index in some corner cases #51954 @lance6716
- Fix the issue that DDL operations get stuck when restoring a table with the foreign key #51838 @YangKeao
- Fix the issue that adding indexes fails when the TiDB network is isolated #51846 @ywqzzy
- Fix the issue that adding an index with the same name after renaming an index causes an error #51431 @lance6716
- Fix the issue of inconsistent data indexes caused by cluster upgrade during adding indexes #52411 @tangenta
- Fix the issue that adding indexes to large tables fails after enabling the Distributed eXecution Framework (DXF) #52640 @tangenta
- Fix the issue that adding indexes concurrently reports the error no such file or directory #52475 @tangenta
- Fix the issue that the temporary data cannot be cleaned up after adding indexes fails #52639 @lance6716
- Fix the issue that the metadata lock fails to prevent DDL operations from executing in the plan cache scenario #51407 @wjhuang2016
- Fix the issue that the IMPORT INTO operation gets stuck when importing a large amount of data #52884 @lance6716
- Fix the issue that TiDB unexpectedly restarts when logging gRPC errors #51301 @guo-shaoge

- Fix the issue that IndexHashJoin outputs redundant data when calculating Anti Left Outer Semi Join #52923 @yibin87
- Fix the incorrect result of the TopN operator in correlated subqueries #52777 @yibin87
- Fix the inaccurate execution time statistics of HashJoin probe #52222 @windtalker
- Fix the issue that using TABLESAMPLE returns incorrect results in static partition pruning mode (tidb_partition_prune_mode='static') #52282 @tangenta
- Fix the issue that TTL is deviated by 1 hour in daylight saving time #51675 @lcwangchao
- Fix the incorrect calculation and display of the number of connections (Connection Count) on the TiDB Dashboard Monitoring page #51889 @YangKeao
- Fix the issue that the status gets stuck when rolling back the partition DDL tasks #51090 @jiyfhust
- Fix the issue that the value of max_remote_stream is incorrect when executing EXPLAIN ANALYZE #52646 @JaySon-Huang
- Fix the issue that querying the TIDB_HOT_REGIONS table might incorrectly return INFORMATION_SCHEMA tables #50810 @Defined2014
- Fix the issue that the EXPLAIN statement might display incorrect column IDs in the result when statistics for certain columns are not fully loaded #52207 @time-and-fate
- Fix the issue that the type returned by the IFNULL function is inconsistent with MySQL #51765 @YangKeao
- Fix the issue that adding a unique index might cause TiDB to panic #52312 @wjhuang2016

• TiKV

- Fix the issue that resolve-ts is blocked when a stale Region peer ignores the GC message #16504 @crazycs520
- Fix the issue that inactive Write Ahead Logs (WALs) in RocksDB might corrupt data #16705 @Connor1996

• PD

- Fix the issue that TSO might get stuck when toggling the PD microservice mode on and off #7849 @JmPotato

  - Fix the issue that the State monitoring metric for DR Auto-Sync does not show any data #7974 @lhy1024
  - Fix the issue that checking binary versions might cause PD panic #7978 @JmPotato
  - Fix the type conversion error that occurs when TTL parameters are parsed #7980 @HuSharp
  - Fix the issue that the Leader fails to transfer when you switch it between two deployed data centers #7992 @TonsnakeLin
  - Fix the issue that PrintErrln in pd-ctl fails to output error messages to stderr #8022 @HuSharp
  - Fix the issue that PD might panic when generating Merge schedules #8049 @nolouch
  - Fix the panic issue caused by GetAdditionalInfo #8079 @HuSharp
  - Fix the issue that the Filter target monitoring metric for PD does not provide scatter range information #8125 @HuSharp
  - Fix the issue that the query result of SHOW CONFIG includes the deprecated configuration item trace-region-flow #7917 @rleungx
  - Fix the issue that the scaling progress is not correctly displayed #7726 @CabinfeverB
- TiFlash
  - Fix the issue that TiFlash might panic when you insert data to columns with invalid default values in non-strict sql_mode #8803 @Lloyd-Pottiger
  - Fix the issue that TiFlash might return transiently incorrect results in high-concurrency read scenarios #8845 @JinheLin
  - Fix the issue that in the disaggregated storage and compute architecture, the disk used_size metric displayed in Grafana is incorrect after you modify the value of the storage.remote.cache.capacity configuration item for TiFlash compute nodes #8920 @JinheLin
  - Fix the issue that TiFlash metadata might become corrupted and cause the process to panic when upgrading a cluster from a version earlier than v6.5.0 to v6.5.0 or later #9039 @JaySon-Huang
  - Fix the issue that in the disaggregated storage and compute architecture, TiFlash might panic when the compute node process is stopped #8860 @guo-shaoge

    - Fix the issue that TiFlash might return an error when executing queries containing virtual generated columns #8787 @guo-shaoge
- Tools
    - Backup & Restore (BR)
        - Fix the issue that BR cannot back up the AUTO_RANDOM ID allocation progress in a union clustered index that contains an AUTO_RANDOM column #52255 @Leavrth
        - Fix the issue that removing a log backup task after it is paused does not immediately restore the GC safepoint #52082 @3pointer
        - Fix a rare issue that special event timing might cause the data loss in log backup #16739 @YuJuncen
        - Fix the issue that the global checkpoint of log backup is advanced ahead of the actual backup file write point due to TiKV restart, which might cause a small amount of backup data loss #16809 @YuJuncen
        - Fix the issue that the confusing information related to --concurrency appears in the log during a full backup #50837 @BornChanger
        - Fix the issue that the Region fetched from PD does not have a Leader when restoring data using BR or importing data using TiDB Lightning in physical import mode #51124 #50501 @Leavrth
        - Fix the issue that after pausing, stopping, and rebuilding the log backup task, the task status is normal, but the checkpoint does not advance #53047 @RidRisR
        - Fix the unstable test case TestClearCache #51671 @zxc111
        - Fix the unstable test case TestGetMergeRegionSizeAndCount #52095 @3pointer
        - Fix the unstable integration test br_tikv_outage #52673 @Leavrth
        - Fix the issue that the test case TestGetTSWithRetry takes too long to execute #52547 @Leavrth
        - Fix the issue that TiKV might panic when resuming a paused log backup task with unstable network connections to PD #17020 @YuJuncen

- TiCDC

    - Fix the issue that calling the API (/api/v2/owner/resign) that evicts the TiCDC owner node causes the TiCDC task to restart unexpectedly #10781 @sdojjy
    - Fix the issue that when the downstream Pulsar is stopped, removing the changefeed causes the normal TiCDC process to get stuck, which causes other changefeed processes to get stuck #10629 @asddongmen
    - Fix the issue that the **Ownership history** panel in Grafana is unstable #10796 @hongyunyan
    - Fix the issue that restarting PD might cause the TiCDC node to restart with an error #10799 @3AceShowHand
    - Fix the issue that high latency in the PD disk I/O causes severe latency in data replication #9054 @asddongmen
    - Fix the issue that the default value of TIMEZONE type is not set according to the correct time zone #10931 @3AceShowHand
    - Fix the issue that DROP PRIMARY KEY and DROP UNIQUE KEY statements are not replicated correctly #10890 @asddongmen
    - Fix the issue that TiCDC fails to execute the Exchange Partition ... With Validation DDL downstream after it is written upstream, causing the changefeed to get stuck #10859 @hongyunyan

- TiDB Lightning

    - Fix the issue that TiDB Lightning reports no database selected during data import due to incompatible SQL statements in the source files #51800 @lance6716
    - Fix the issue that TiDB Lightning might print sensitive information to logs in server mode #36374 @kennytm
    - Fix the issue that killing the PD Leader causes TiDB Lightning to report the invalid store ID 0 error during data import #50501 @Leavrth
    - Fix the issue that TiDB Lightning reports the Unknown column in where clause error when processing conflicting data using the replace strategy #52886 @lyzx2001

- Fix the issue that TiDB Lightning panics when importing an empty table of Parquet format #52518 @kennytm

## 1.6 Performance test

To learn about the performance of TiDB v8.1.0, you can refer to the TPC-C performance test report and Sysbench performance test report of the TiDB Dedicated cluster.

## 1.7 Contributors

We would like to thank the following contributors from the TiDB community:

- arturmelanchyk (First-time contributor)
- CabinfeverB
- danqixu (First-time contributor)
- imalasong (First-time contributor)
- jiyfhust
- negachov (First-time contributor)
- testwill
- yzhan1 (First-time contributor)
- zxc111 (First-time contributor)

# 2 TiDB 8.0.0 Release Notes

Release date: March 29, 2024

TiDB version: 8.0.0

Quick access: Quick start

8.0.0 introduces the following key features and improvements:

| Category | Feature/Enhancement | Description |
|---|---|---|
| Scalability and Performance | Disaggregation of PD to improve scalability (experimental) | Placement Driver (PD) contains multiple critical modules to ensure the normal operation of TiDB clusters. As the workload of a cluster increases, the resource consumption of each module in PD also increases, causing mutual interference between these modules and ultimately affecting the overall service quality of the cluster. Starting from v8.0.0, TiDB addresses this issue by splitting the TSO and scheduling modules in PD into |

| Category | Feature/Enhancement | Description |
|---|---|---|
| | | independently deployable microservices. This can significantly reduce the mutual interference between modules as the cluster scales. With this architecture, much larger clusters with much larger workloads are now possible. |
| | Bulk DML for much larger transactions (experimental) | Large batch DML jobs, such as extensive cleanup jobs, joins, or aggregations, can consume a significant amount of memory and have previously been limited at very large scales. Bulk DML (tidb_dml_type = "bulk") is a new DML type for handling large batch DML tasks more efficiently while providing transaction guarantees and mitigating OOM issues. This feature differs from import, load, and restore operations when used for data loading. |
| | Acceleration of cluster snapshot restore speed (GA) | With this feature, BR can fully leverage the scale advantage of a cluster, enabling all TiKV nodes in the cluster to participate in the preparation step of data restores. This feature can significantly improve the restore speed of large datasets in large-scale clusters. Real-world tests show that this feature can saturate the download bandwidth, with the download speed improving by 8 to 10 times, and the end-to-end restore speed improving by approximately 1.5 to 3 times. |
| | Enhance the stability of caching the schema information when there is a massive number of tables (experimental) | SaaS companies using TiDB as the system of record for their multi-tenant applications often need to store a substantial number of tables. In previous versions, handling table counts in the order of a million or more was feasible, but it had the potential to degrade the overall user experience. TiDB v8.0.0 improves the situation with the following enhancements:<br>• Introduce a new schema information caching system, incorporating a lazy-loading Least Recently Used (LRU) cache for table metadata and more efficiently managing schema version changes. |

| Category | Feature/Enhancement | Description |
|---|---|---|
| | | • Implement a priority queue for auto analyze, making the process less rigid and enhancing stability across a wider array of tables. |
| DB Operations and Observability | Support monitoring index usage statistics | Proper index design is a crucial prerequisite to maintaining database performance. TiDB v8.0.0 introduces the INFORMATION_SCHEMA.TIDB_INDEX_USAGE table and the sys.schema_unused_indexes view to provide usage statistics of indexes. This feature helps you assess the efficiency of indexes in the database and optimize the index design. |
| Data Migration | TiCDC adds support for the Simple protocol | TiCDC introduces a new protocol, the Simple protocol. This protocol provides in-band schema tracking capabilities by embedding table schema information in DDL and BOOTSTRAP events. |
| | TiCDC adds support for the Debezium format protocol | TiCDC introduces a new protocol, the Debezium protocol. TiCDC can now publish data change events to a Kafka sink using a protocol that generates Debezium style messages. |

## 2.1 Feature details

### 2.1.1 Scalability

- PD supports the microservice mode (experimental) #5766 @binshi-bing

  Starting from v8.0.0, PD supports the microservice mode. This mode splits the timestamp allocation and cluster scheduling functions of PD into separate microservices that can be deployed independently, thereby enhancing performance scalability for PD and addressing performance bottlenecks of PD in large-scale clusters.

  - tso microservice: provides monotonically increasing timestamp allocation for the entire cluster.
  - scheduling microservice: provides scheduling functions for the entire cluster, including but not limited to load balancing, hot spot handling, replica repair, and replica placement.

Each microservice is deployed as an independent process. If you configure more than one replica for a microservice, the microservice automatically implements a primary-secondary fault-tolerant mode to ensure high availability and reliability of the service.

Currently, PD microservices can only be deployed using TiDB Operator. It is recommended to consider this mode when PD becomes a significant performance bottleneck that cannot be resolved by scaling up.

For more information, see documentation.

- Enhance the usability of the Titan engine #16245 @Connor1996
  - Enable the shared cache for Titan blob files and RocksDB block files by default (shared-blob-cache defaults to true), eliminating the need to configure blob-cache-size separately.
  - Support dynamically modifying min-blob-size, blob-file-compression, and discardable-ratio to improve performance and flexibility when using the Titan engine.

  For more information, see documentation.

### 2.1.2 Performance

- BR improves snapshot restore speed (GA) #50701 @3pointer @Leavrth

  Starting from TiDB v8.0.0, the acceleration of snapshot restore speed is now generally available (GA) and enabled by default. BR improves the snapshot restore speed significantly by implementing various optimizations such as adopting the coarse-grained Region scattering algorithm, creating databases and tables in batches, reducing the mutual impact between SST file downloads and ingest operations, and accelerating the restore of table statistics. According to test results from real-world cases, the data restore speed of a single TiKV node stabilizes at 1.2 GiB/s, and 100 TiB of data can be restored within one hour.

  This means that even in high-load environments, BR can fully utilize the resources of each TiKV node, significantly reducing database restore time, enhancing the availability and reliability of databases, and reducing downtime and business losses caused by data loss or system failures. Note that the increase in restore speed is attributed to the parallel execution of a large number of goroutines, which can result in significant memory consumption, especially when there are many tables or Regions. It is recommended to use machines with higher memory capacity to run the BR client. If the memory capacity of the machine is limited, it is recommended to use a finer-grained Region scattering algorithm. In addition, because the coarse-grained Region scattering algorithm might consume a significant amount of external storage bandwidth, you need to avoid any impact on other applications due to insufficient external bandwidth.

For more information, see documentation.

- Support pushing down the following functions to TiFlash #50975 #50485 @yibin87 @windtalker

    - CAST(DECIMAL AS DOUBLE)
    - POWER()

    For more information, see documentation.

- The concurrent HashAgg algorithm of TiDB supports disk spill (experimental) #35637 @xzhangxian1008

    In earlier versions of TiDB, the concurrency algorithm of the HashAgg operator does not support disk spill. If the execution plan of a SQL statement contains the concurrent HashAgg operator, all the data for that SQL statement can only be processed in memory. Consequently, TiDB has to process a large amount of data in memory. When the data size exceeds the memory limit, TiDB can only choose the non-concurrent algorithm, which does not leverage concurrency for performance improvement.

    In v8.0.0, the concurrent HashAgg algorithm of TiDB supports disk spill. Under any concurrent conditions, the HashAgg operator can automatically trigger data spill based on memory usage, thus balancing performance and data throughput. Currently, as an experimental feature, TiDB introduces the tidb_enable_concurrent_hashagg_spill variable to control whether to enable the concurrent HashAgg algorithm that supports disk spill. When this variable is ON, it means enabled. This variable will be deprecated after the feature is generally available in a future release.

For more information, see documentation.

- Introduce the priority queue for automatic statistics collection #50132 @hi-rustin

  Maintaining optimizer statistics up-to-date is the key to stabilizing database performance. Most users rely on the automatic statistics collection provided by TiDB to collect the latest statistics. Automatic statistics collection checks the status of statistics for all objects, and adds unhealthy objects to a queue for sequential collections. In previous versions, the order is random, which could result in excessive waits for more worthy candidates to be updated, causing potential performance regressions.

  Starting from v8.0.0, automatic statistics collection dynamically sets priorities for objects in combination with a variety of conditions to ensure that more deserving candidates are processed in priority, such as newly created indexes and partitioned tables with definition changes. Additionally, TiDB prioritizes tables with lower health scores, placing them at the top of the queue. This enhancement makes the order of collection more reasonable, and reduces performance problems caused by outdated statistics, therefore improving database stability.

  For more information, see documentation.

- Remove some limitations on execution plan cache #49161 @mjonss @qw4990

  TiDB supports plan cache, which can effectively reduce the latency of OLTP systems and is important for performance. In v8.0.0, TiDB removes several limitations on plan cache. Execution plans with the following items can be cached now:

  - Partitioned tables
  - Generated columns, including objects that depend on generated columns (such as multi-valued indexes)

  This enhancement extends the use cases of plan cache and improves the overall database performance in complex scenarios.

  For more information, see documentation.

- Optimizer enhances support for multi-valued indexes #47759 #46539 @Arenatlx @time-and-fate

  TiDB v6.6.0 introduces multi-value indexes to improve query performance for JSON data types. In v8.0.0, the optimizer enhances its support for multi-valued indexes and can correctly identify and utilize them to optimize queries in complex scenarios.

  - The optimizer collects statistics on multi-valued indexes and decides execution plans with the statistics. If several multi-value indexes can be selected by a SQL statement, the optimizer can identify the one with lower cost.
  - When using OR to connect multiple member of conditions, the optimizer can match an effective index partial path for each DNF item (a member of condition) and combine these paths using Union to form an Index Merge. This achieves more efficient condition filtering and data fetch.

  For more information, see documentation.

- Support configuring the update interval for low-precision TSO #51081 @Tema

  The low-precision TSO feature in TiDB uses regularly updated TSO as the transaction timestamp. In scenarios where reading outdated data is acceptable, this feature reduces the overhead of obtaining TSO for small read-only transactions by sacrificing real-time performance and improves the ability of high-concurrency reads.

  Before v8.0.0, the TSO update interval of low-precision TSO feature is fixed and cannot be adjusted according to actual application requirements. In v8.0.0, TiDB introduces the system variable tidb_low_resolution_tso_update_interval to control the TSO update interval. This feature takes effect only when the low-precision TSO feature is enabled.

  For more information, see documentation.

### 2.1.3 Reliability

- Support caching required schema information according to the LRU algorithm to reduce memory consumption on the TiDB server (experimental) #50959 @gmhdbjd

Before v8.0.0, each TiDB node caches the schema information of all tables. In scenarios with hundreds of thousands of tables, just caching these table schemas could consume a significant amount of memory.

Starting from v8.0.0, TiDB introduces the system variable tidb_schema_cache_size, which enables you to set an upper limit for caching schema information, thereby preventing excessive memory usage. When you enable this feature, TiDB uses the Least Recently Used (LRU) algorithm to cache the required tables, effectively reducing the memory consumed by the schema information.

For more information, see documentation.

### 2.1.4 Availability

- The proxy component TiProxy becomes generally available (GA) #413 @djshow832 @xhebox

  TiDB v7.6.0 introduces the proxy component TiProxy as an experimental feature. TiProxy is the official proxy component of TiDB, located between the client and TiDB server. It provides load balancing and connection persistence functions for TiDB, making the workload of the TiDB cluster more balanced and not affecting user access to the database during maintenance operations.

  In v8.0.0, TiProxy becomes generally available and enhances the automatic generation of signature certificates and monitoring functions.

  The usage scenarios of TiProxy are as follows:

  - During maintenance operations such as rolling restarts, rolling upgrades, and scaling-in in a TiDB cluster, changes occur in the TiDB servers which result in interruptions in connections between clients and the TiDB servers. By using TiProxy, connections can be smoothly migrated to other TiDB servers during these maintenance operations so that clients are not affected.
  - Client connections to a TiDB server cannot be dynamically migrated to other TiDB servers. When the workload of multiple TiDB servers is unbalanced, it might result in a situation where the overall cluster resources are sufficient, but certain TiDB

servers experience resource exhaustion leading to a significant increase in latency. To address this issue, TiProxy provides dynamic migration for connection, which allows connections to be migrated from one TiDB server to another without any impact on the clients, thereby achieving load balancing for the TiDB cluster.

TiProxy has been integrated into TiUP, TiDB Operator, and TiDB Dashboard, making it easy to configure, deploy, and maintain.

For more information, see documentation.

### 2.1.5 SQL

- Support a new DML type for handling a large amount of data (experimental) #50215 @ekexium

  Before v8.0.0, TiDB stores all transaction data in memory before committing. When processing a large amount of data, the memory required for transactions becomes a bottleneck that limits the transaction size that TiDB can handle. Although TiDB introduces non-transactional DML to attempt to solve the transaction size limitation by splitting SQL statements, this feature has various limitations and does not provide an ideal experience in actual scenarios.

  Starting from v8.0.0, TiDB supports a DML type for handling a large amount of data. This DML type writes data to TiKV in a timely manner during execution, avoiding the continuous storage of all transaction data in memory, and thus supports handling a large amount of data that exceeds the memory limit. This DML type ensures transaction integrity and uses the same syntax as standard DML. INSERT, UPDATE, REPLACE, and DELETE statements can use this new DML type to execute large-scale DML operations.

  This DML type is implemented by the Pipelined DML feature and only takes effect on statements with auto-commit enabled. You can control whether to enable this DML type by setting the system variable tidb_dml_type.

  For more information, see documentation.

- Support using some expressions to set default column values when creating a table (experimental) #50936 @zimulala

Before v8.0.0, when you create a table, the default value of a column is limited to strings, numbers, and dates. Starting from v8.0.0, you can use some expressions as the default column values. For example, you can set the default value of a column to UUID(). This feature helps you meet more diverse requirements.

For more information, see documentation.

- Support the div_precision_increment system variable #51501 @yibin87

  MySQL 8.0 supports the variable div_precision_increment, which specifies the number of digits by which to increase the scale of the result of a division operation performed using the / operator. Before v8.0.0, TiDB does not support this variable, and division is performed to 4 decimal places. Starting from v8.0.0, TiDB supports this variable. You can specify the number of digits by which to increase the scale of the result of a division operation as desired.

  For more information, see documentation.

### 2.1.6 DB operations

- PITR supports Amazon S3 Object Lock #51184 @RidRisR

  Amazon S3 Object Lock can help prevent backup data from accidental or intentional deletion during a specified retention period, enhancing the security and integrity of data. Starting from v6.3.0, BR supports Amazon S3 Object Lock for snapshot backups, adding an additional layer of security for full backups. Starting from v8.0.0, PITR also supports Amazon S3 Object Lock. Whether for full backups or log data backups, the Object Lock feature ensures more reliable data protection, further strengthening the security of data backup and recovery and meeting regulatory requirements.

  For more information, see documentation.

- Support making invisible indexes visible at the session level #50653 @hawkingrei

  By default, the optimizer does not select invisible indexes. This mechanism is usually used to evaluate whether to delete an index. If there is uncertainty about the potential performance impact of deleting an index, you have the option to set the index to invisible temporarily and promptly restore it to visible when needed.

Starting from v8.0.0, you can set the session-level system variable tidb_opt_use_invisible_indexes to ON to make the current session aware of invisible indexes. With this feature, you can create a new index and test its performance by making the index visible first, and then modifying the system variable in the current session for testing without affecting other sessions. This improvement enhances the safety of SQL tuning and helps to improve the stability of production databases.

For more information, see documentation.

- Support writing general logs to a separate file #51248 @Defined2014

  The general log is a MySQL-compatible feature that logs all executed SQL statements to help diagnose issues. TiDB also supports this feature. You can enable it by setting the variable tidb_general_log. However, in previous versions, the content of general logs can only be written to the TiDB instance log along with other information, which is inconvenient for users who need to keep logs for a long time.

  Starting from v8.0.0, you can write the general log to a specified file by setting the configuration item log.general-log-file to a valid filename. The general log follows the same rotation and retention policies as the instance log.

  In addition, to reduce the disk space occupied by historical log files, TiDB v8.0.0 introduces a native log compression option. You can set the configuration item log.file.compression to gzip to automatically compress rotated logs using the gzip format.

  For more information, see documentation.

### 2.1.7 Observability

- Support monitoring index usage statistics #49830 @YangKeao

  Proper index design is a crucial prerequisite to maintaining database performance. TiDB v8.0.0 introduces the INFORMATION_SCHEMA.TIDB_INDEX_USAGE table, which records the statistics of all indexes on the current TiDB node, including the following information:

- The cumulative execution count of statements that scan the index
- The total number of rows scanned when accessing the index
- The selectivity distribution when scanning the index
- The time of the most recent access to the index

With this information, you can identify indexes that are not used by the optimizer and indexes with poor selectivity, thereby optimizing index design to improve database performance.

Additionally, TiDB v8.0.0 introduces a view sys.schema_unused_indexes, which is compatible with MySQL. This view shows indexes that have not been used since the last start of TiDB instances. For clusters upgraded from versions earlier than v8.0.0, the sys schema and the views are not created automatically. You can manually create them by referring to sys.schema_unused_indexes.

For more information, see documentation.

### 2.1.8 Security

- TiKV encryption at rest supports Google Key Management Service (Cloud KMS) (experimental) #8906 @glorv

  TiKV ensures data security by encrypting stored data using the encryption at rest technique. The core of encryption at rest for security is key management. Starting from v8.0.0, you can manage the master key of TiKV using Google Cloud KMS to establish encryption-at-rest capabilities based on Cloud KMS, thereby enhancing the security of user data.

  To enable encryption at rest based on Google Cloud KMS, you need to create a key on Google Cloud and then configure the [security.encryption.master-key] section in the TiKV configuration file.

  For more information, see documentation.

- Enhance TiDB log desensitization #51306 @xhebox

  The enhancement of TiDB log desensitization is based on marking SQL text information in log files, facilitating the safe display of sensitive data when users view the logs. You can control whether to desensitize log information to enable secure use of TiDB logs in different scenarios, enhancing the security and flexibility of using log desensitization. To use this feature, set the system variable tidb_redact_log to MARKER. This marks the SQL text in TiDB logs. When you view the logs, sensitive data is securely displayed based on the markers, thus protecting the log information.

For more information, see documentation.

### 2.1.9 Data migration

- TiCDC adds support for the Simple protocol #9898 @3AceShowHand

  TiCDC introduces a new protocol, the Simple protocol. This protocol provides in-band schema tracking capabilities by embedding table schema information in DDL and BOOTSTRAP events.

  For more information, see documentation.

- TiCDC adds support for the Debezium format protocol #1799 @breezewish

  TiCDC can now publish data change events to a Kafka sink using a protocol that generates event messages in a Debezium style format. This helps to simplify the migration from MySQL to TiDB for users who are currently using Debezium to pull data from MySQL for downstream processing.

  For more information, see documentation.

- DM supports using a user-provided secret key to encrypt and decrypt passwords of source and target databases #9492 @D3Hunter

  In earlier versions, DM uses a built-in fixed secret key with relatively low security. Starting from v8.0.0, you can upload and specify a secret key file for encrypting and decrypting passwords of upstream and downstream databases. In addition, you can replace the secret key file as needed to enhance data security.

  For more information, see documentation.

- Supports the IMPORT INTO ... FROM SELECT syntax to enhance the IMPORT INTO functionality (experimental) #49883 @D3Hunter

In earlier TiDB versions, importing query results into a target table could only be done using the INSERT INTO ... SELECT statement, which is relatively inefficient in some large dataset scenarios. Starting from v8.0.0, TiDB enables you to use IMPORT INTO ... FROM SELECT to import the results of a SELECT query into an empty TiDB target table, which achieves up to 8 times the performance of INSERT INTO ... SELECT and significantly reduces the import time.

In addition, you can use IMPORT INTO ... FROM SELECT to import historical data queried with AS OF TIMESTAMP.

For more information, see documentation.

- TiDB Lightning simplifies conflict resolution strategies and supports handling conflicting data using the replace strategy (experimental) #51036 @lyzx2001

  In earlier versions, TiDB Lightning has one data conflict resolution strategy for the logical import mode and two data conflict resolution strategies for the physical import mode, which are not easy to understand and configure.

  Starting from v8.0.0, TiDB Lightning deprecates the old version of conflict detection strategy for the physical import mode, enables you to control the conflict detection strategy for both logical and physical import modes via the conflict.strategy parameter, and simplifies the configuration of this parameter. In addition, in the physical import mode, the replace strategy now supports retaining the latest data and overwriting the old data when the import detects data with primary key or unique key conflicts.

  For more information, see documentation.

- Global Sort becomes generally available (GA), improving the performance and stability of IMPORT INTO significantly #45719 @lance6716

  Before v7.4.0, when executing IMPORT INTO tasks using the Distributed eXecution Framework (DXF), TiDB only locally sorts part of the data before importing it into TiKV due to limited local storage space. This results in significant overlap of the imported data in TiKV, requiring TiKV to perform additional compaction operations during import and affecting the TiKV performance and stability.

With the Global Sort experimental feature introduced in v7.4.0, TiDB can temporarily store the data to be imported in an external storage (such as Amazon S3) for global sorting before importing it into TiKV, which eliminates the need for TiKV compaction operations during import. In v8.0.0, Global Sort becomes GA. This feature reduces the resource consumption of TiKV and significantly improves the performance and stability of IMPORT INTO. If you enable the Global Sort, each IMPORT INTO task supports importing data within 40 TiB.

For more information, see documentation.

## 2.2 Compatibility changes

**Note:**

This section provides compatibility changes you need to know when you upgrade from v7.6.0 to the current version (v8.0.0). If you are upgrading from v7.5.0 or earlier versions to the current version, you might also need to check the compatibility changes introduced in intermediate versions.

- Remove witness-related schedulers that are not GA but are enabled by default #7765 @rleungx

### 2.2.1 Behavior changes

- Prohibit setting require_secure_transport to ON in Security Enhanced Mode (SEM) to prevent potential connectivity issues for users. #47665 @tiancaiamao
- DM removes the fixed secret key for encryption and decryption and enables you to customize a secret key for encryption and decryption. If encrypted passwords are used in data source configurations and migration task configurations before the upgrade, you need to refer to the upgrade steps in Customize a Secret Key for DM Encryption and Decryption for additional operations. #9492 @D3Hunter
- Before v8.0.0, after enabling the acceleration of ADD INDEX and CREATE INDEX (tidb_ddl_enable_fast_reorg = ON), the encoded index key ingests data to TiKV with a fixed concurrency of 16, which cannot be dynamically adjusted according to the downstream TiKV capacity. Starting from v8.0.0, you can adjust the concurrency using the

tidb_ddl_reorg_worker_cnt system variable. The default value is 4. Compared with the previous default value of 16, the new default value reduces performance when ingesting indexed key-value pairs. You can adjust this system variable based on the workload of your cluster.

### 2.2.2 MySQL compatibility

- The KEY partition type supports statements with an empty list of partition fields, which is consistent with the behavior of MySQL.

### 2.2.3 System variables

| Variable name | Change type | Description |
|---|---|---|
| tidb_disable_txn_auto_retry | Deprecated | Starting from v8.0.0, this system variable is deprecated, and TiDB no longer supports automatic retries of optimistic transactions. It is recommended to use the Pessimistic transaction mode. If you encounter optimistic transaction conflicts, you can capture the error and retry transactions in your application. |
| tidb_ddl_version | Renamed | Controls whether to enable TiDB DDL V2. Starting from v8.0.0, this variable is renamed to tidb_enable_fast_create_table to better reflect its purpose. |
| tidb_enable_collect_execution_info | Modified | Adds a control to whether to record the usage statistics of indexes. The default value is ON. |
| tidb_redact_log | Modified | Controls how to handle user information in SAL text when logging TiDB logs and slow logs. The value options are OFF (indicating not processing user information in the log) and ON (indicating hiding user information in the log). To provide a richer way of processing user information in the log, the MARKER option is added in v8.0.0 to support marking log information. |
| div_precision_increment | Newly added | Controls the number of digits by which to increase the scale of the result of a division operation performed using the / operator. This variable is the same as MySQL. |
| tidb_dml_type | Newly added | Controls the execution mode of DML statements. The value options are "standard" and "bulk". |
| tidb_enable_auto_analyze_priority_queue | Newly added | Controls whether to enable the priority queue to schedule the tasks of automatically collecting statistics. When this variable is enabled, TiDB prioritizes collecting statistics for the tables that most need statistics. |

| Variable name | Change type | Description |
|---|---|---|
| tidb_enable_concurrent_hashagg_spill | Newly added | Controls whether TiDB supports disk spill for the concurrent HashAgg algorithm. When it is ON, disk spill can be triggered for the concurrent HashAgg algorithm. This variable will be deprecated when this feature is generally available in a future release. |
| tidb_enable_fast_create_table | Newly added | Controls whether to enable TiDB Accerates Table Creation. Set the value to ON to enable it and OFF to disable it. The default value is ON. When this variable is enabled, TiDB accelerates table creation by using CREATE TABLE. |
| tidb_load_binding_timeout | Newly added | Controls the timeout of loading bindings. If the execution time of loading bindings exceeds this value, the loading will stop. |
| tidb_low_resolution_tso_update_interval | Newly added | Controls the interval for updating TiDB cache timestamp. |
| tidb_opt_ordering_index_selectivity_ratio | Newly added | Controls the estimated number of rows for an index that matches the SQL statement ORDER BY when there are ORDER BY and LIMIT clauses in a SQL statement, but some filter conditions not covered by the index. The default value is -1, which means to disable this system variable. |
| tidb_opt_use_invisible_indexes | Newly added | Controls whether the optimizer can select invisible indexes for query optimization in the current session. When the variable is set to ON, the optimizer can select invisible indexes for query optimization in the session. |
| tidb_schema_cache_size | Newly added | Controls the upper limit of memory that can be used for caching the schema information to avoid occupying too much memory. When this feature is enabled, the LRU algorithm is used to cache the required tables, effectively reducing the memory occupied by the schema information. |

### 2.2.4 Configuration file parameters

| Configuration file | Configuration parameter | Change type | Description |
|---|---|---|---|
| TiDB | instance.tidb_enable_collect_execution_info | Modified | Adds a control to whether to record the usage statistics of indexes. The default value is true. |

| Configuration file | Configuration parameter | Change type | Description |
|---|---|---|---|
| TiDB | tls-version | Modified | This parameter no longer supports "TLSv1.0" and "TLSv1.1". Now it only supports "TLSv1.2" and "TLSv1.3". |
| TiDB | log.file.compression | Newly added | Specifies the compression format of the polling log. The default value is null, which means that the polling log is not compressed. |
| TiDB | log.general-log-file | Newly added | Specifies the file to save the general log to. The default is null, which means that the general log will be written to the instance file. |
| TiDB | tikv-client.enable-replica-selector-v2 | Newly added | Controls whether to use the new version of the Region replica selector when sending RPC requests to TiKV. The default value is true. |
| TiKV | log-backup.initial-scan-rate-limit | Modified | Adds a limit of 1MiB as the minimum value. |
| TiKV | raftstore.store-io-pool-size | Modified | Changes the default value from 0 to 1 to improve TiKV performance, meaning that the size of the StoreWriter thread pool now defaults to 1. |
| TiKV | rocksdb.defaultcf.titan.blob-cache-size | Modified | Starting from v8.0.0, TiKV introduces the shared-blob-cache configuration item and enables it by default, so there is no need to set blob-cache-size separately. The configuration of blob-cache-size only takes effect when shared-blob-cache is set to false. |
| TiKV | security.encryption.master-key.vendor | Modified | Adds gcp as an available type for the service provider. |
| TiKV | rocksdb.defaultcf.titan.shared-blob-cache | Newly added | Controls whether to enable the shared cache for Titan blob files and RocksDB block files. The default value is true. |

| Configuration file | Configuration parameter | Change type | Description |
|---|---|---|---|
| TiKV | security.encryption.master-key.gcp.credential-file-path | Newly added | Specifies the path to the Google Cloud authentication credentials file when security.encryption.master-key.vendor is gcp. |
| TiDB Lightning | tikv-importer.duplicate-resolution | Deprecated | Controls whether to detect and resolve unique key conflicts in physical import mode. Starting from v8.0.0, it is replaced by conflict.strategy. |
| TiDB Lightning | conflict.precheck-conflict-before-import | Newly added | Controls whether to enable preprocess conflict detection, which checks conflicts in data before importing it to TiDB. The default value of this parameter is false, which means that TiDB Lightning only checks conflicts after the data import. This parameter can be used only in the physical import mode (tikv-importer.backend = "local"). |
| TiDB Lightning | logical-import-batch-rows | Newly added | Controls the maximum number of rows inserted per transaction in the logical import mode. The default value is 65536 rows. |
| TiDB Lightning | logical-import-batch-size | Newly added | Controls the maximum size of each SQL query executed on the downstream TiDB server in the logical import mode. The default value is "96KiB". The unit can be KB, KiB, MB, or MiB. |
| Data Migration | secret-key-path | Newly added | Specifies the file path of the secret key, which is used to encrypt and decrypt upstream and downstream passwords. The file must contain a 64-character hexadecimal AES-256 secret key. |
| TiCDC | tls-certificate-file | Newly added | Specifies the path to the encrypted certificate file on the client, which is required when Pulsar enables TLS encrypted transmission. |
| TiCDC | tls-key-file-path | Newly added | Specifies the path to the encrypted private key on the client, which is required when Pulsar enables TLS encrypted transmission. |

#### 2.2.5 System tables

- Add new system tables INFORMATION_SCHEMA.TIDB_INDEX_USAGE and INFORMATION_SCHEMA.CLUSTER_TIDB_INDEX_USAGE to record index usage statistics on TiDB nodes.
- Add a new system schema sys and a new view sys.schema_unused_indexes, which records indexes that have not been used since the last start of TiDB.

### 2.3 Deprecated features

- Starting from v8.0.0, the tidb_disable_txn_auto_retry system variable is deprecated, and TiDB no longer supports automatic retries of optimistic transactions. As an alternative, when encountering optimistic transaction conflicts, you can capture the error and retry transactions in your application, or use the Pessimistic transaction mode instead.
- Starting from v8.0.0, TiDB no longer supports the TLSv1.0 and TLSv1.1 protocols. You must upgrade TLS to TLSv1.2 or TLSv1.3.
- Starting from v8.0.0, TiDB Lightning deprecates the old version of conflict detection strategy for the physical import mode, and enables you to control the conflict detection strategy for both logical and physical import modes via the conflict.strategy parameter. The duplicate-resolution parameter for the old version of conflict detection will be removed in a future release.
- It is planned to redesign the auto-evolution of execution plan bindings in subsequent releases, and the related variables and behavior will change.

### 2.4 Improvements

- TiDB
  - Improve the performance of executing the CREATE TABLE DDL statement by 10 times and support linear scalability #50052 @GMHDBJD
  - Support submitting 16 IMPORT INTO ... FROM FILE tasks simultaneously, facilitating bulk data import into target tables and significantly improving the efficiency and performance of importing data files #49008 @D3Hunter
  - Improve the performance of spilling data to disk for the Sort operator #47733 @xzhangxian1008

- Support canceling queries during spilling data to disk, which optimizes the exit mechanism of the data spill feature #50511 @wshwsh12
- Support using an index that matches partial conditions to construct Index Join when processing table join queries with multiple equal conditions #47233 @winoros
- Enhance the capability of Index Merge to identify sorting requirements in queries and select indexes that meet the sorting requirements #48359 @AilinKid
- When the Apply operator is not executed concurrently, TiDB enables you to view the name of the operator that blocks the concurrency by executing SHOW WARNINGS #50256 @hawkingrei
- Optimize the index selection for point get queries by selecting the most optimal index for queries when all indexes support point get queries #50184 @elsa0520
- Temporarily adjust the priority of statistics synchronously loading tasks to high to avoid widespread timeouts during TiKV high loads, as these timeouts might result in statistics not being loaded #50332 @winoros
- When the PREPARE statement fails to hit the execution plan cache, TiDB enables you to view the reason by executing SHOW WARNINGS #50407 @hawkingrei
- Improve the accuracy of query estimation information when the same row of data is updated multiple times #47523 @terry1purcell
- Index Merge supports embedding multi-value indexes and OR operators in AND predicates #51778 @time-and-fate
- When force-init-stats is set to true, TiDB waits for statistics initialization to finish before providing services during TiDB startup. This setting no longer blocks the startup of HTTP servers, which enables users to continue monitoring #50854 @hawkingrei
- MemoryTracker can track the memory usage of the IndexLookup operator #45901 @solotzg
- MemoryTracker can track the memory usage of the MemTableReaderExec operator #51456 @wshwsh12

    - Support loading Regions in batch from PD to speed up the conversion process from the KV range to Regions when querying large tables #51326 @SeaRise
    - Optimize the query performance of the system tables INFORMATION_SCHEMA.TABLES, INFORMATION_SCHEMA.STATISTICS, INFORMATION_SCHEMA.KEY_COLUMN_USAGE, and INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS. Compared with earlier versions, the performance has been improved by up to 100 times. #50305 @ywqzzy

- TiKV

    - Enhance TSO verification and detection to improve the robustness of the cluster TSO when the configuration or operation is improper #16545 @cfzjywxk
    - Optimize the logic of cleaning up pessimistic locks to improve the processing performance of uncommitted transactions #16158 @cfzjywxk
    - Introduce unified health control for TiKV to reduce the impact of abnormal single TiKV node on cluster access performance. You can disable this optimization by setting tikv-client.enable-replica-selector-v2 to false. #16297 #1104 #1167 @MyonKeminta @zyguan @crazycs520
    - The PD client uses the metadata storage interface to replace the previous global configuration interface #14484 @HuSharp
    - Enhance the scanning performance by determining the data loading behavior through write cf stats #16245 @Connor1996
    - Check the latest heartbeat for nodes being deleted and voters being demoted during the Raft conf change process to ensure that this behavior does not make the Region inaccessible #15799 @tonyxuqqi
    - Add Flush and BufferBatchGet interfaces for Pipelined DML #16291 @ekexium
    - Add monitoring and alerting for cgroup CPU and memory limits #16392 @pingandb
    - Add CPU monitoring for Region workers and snapshot generation workers #16562 @Connor1996

    - Add slow logs for peer and store messages #16600 @Connor1996
- PD
    - Enhance the service discovery capability of the PD client to improve its high availability and load balancing #7576 @CabinfeverB
    - Enhance the retry mechanism of the PD client #7673 @JmPotato
    - Add monitoring and alerting for cgroup CPU and memory limits #7716 #7918 @pingandb @rleungx
    - Improve the performance and high availability when using etcd watch #7738 #7724 #7689 @lhy1024
    - Add more monitoring metrics for heartbeat to better analyze performance bottlenecks #7868 @nolouch
    - Reduce the impact of the etcd leader on the PD leader #7499 @JmPotato @HuSharp
    - Enhance the detection mechanism for unhealthy etcd nodes #7730 @JmPotato @HuSharp
    - Optimize the output of GC safepoint in pd-ctl #7767 @nolouch
    - Support dynamic modification of the historical window configuration in the hotspot scheduler #7877 @lhy1024
    - Reduce the lock contention issue in creating operators #7837 @Leavrth
    - Adjust GRPC configurations to improve availability #7821 @rleungx
- TiFlash
    - Support using non-constant values for the json_path argument in the JSON_EXTRACT() function #8510 @SeaRise
    - Support the JSON_LENGTH(json, path) function #8711 @SeaRise
- Tools
    - Backup & Restore (BR)
        - Introduce a new restore parameter --load-stats for the br command-line tool, which controls whether to restore statistics #50568 @Leavrth

    - Introduce a new restore parameter --tikv-max-restore-concurrency for the br command-line tool, which controls the maximum number of download and ingest files for each TiKV node. This parameter also controls the memory consumption of a BR node by controlling the maximum length of the job queue. #51621 @3pointer
    - Enhance restore performance by enabling the coarse-grained Region scatter algorithm to adaptively obtain concurrent parameters #50701 @3pointer
    - Display the log command in the command-line help information of br #50927 @RidRisR
    - Support pre-allocating Table ID during the restore process to maximize the reuse of Table ID and improve restore performance #51736 @Leavrth
    - Disable the GC memory limit tuner feature within TiDB when using BR to avoid OOM issues #51078 @Leavrth
    - Improve the speed of merging SST files during data restore by using a more efficient algorithm #50613 @Leavrth
    - Support creating databases in batch during data restore #50767 @Leavrth
    - Print the information of the slowest Region that affects global checkpoint advancement in logs and metrics during log backups #51046 @YuJuncen
    - Improve the table creation performance of the RESTORE statement in scenarios with large datasets #48301 @Leavrth
- TiCDC
    - Optimize the memory consumption of RowChangedEvent to reduce memory consumption when TiCDC replicates data #10386 @lidezhu
    - Verify that the start-ts parameter is valid when creating and resuming a changefeed task #10499 @3AceShowHand
- TiDB Data Migration (DM)
    - In a MariaDB primary-secondary replication scenario, where the migration path is: MariaDB primary instance ->

MariaDB secondary instance -> DM -> TiDB, when gtid_strict_mode = off and the GTID of the MariaDB secondary instance is not strictly incrementing (for example, there is data writing to the MariaDB secondary instance), the DM task will report an error less than global checkpoint position. Starting from v8.0.0, TiDB is compatible with this scenario and data can be migrated downstream normally. #10741 @okJiang

- TiDB Lightning
  - Support configuring the maximum number of rows in a batch in logical import mode using logical-import-batch-rows #46607 @kennytm
  - TiDB Lightning reports an error when the space of TiFlash is insufficient #50324 @okJiang

## 2.5 Bug fixes

- TiDB
  - Fix the issue that auto analyze is triggered multiple times when there is no data change #51775 @hi-rustin
  - Fix the issue that the auto analyze concurrency is set incorrectly #51749 @hawkingrei
  - Fix the issue of index inconsistency caused by adding multiple indexes using a single SQL statement #51746 @tangenta
  - Fix the Column ... in from clause is ambiguous error that might occur when a query uses NATURAL JOIN #32044 @AilinKid
  - Fix the issue of wrong query results due to TiDB incorrectly eliminating constant values in group by #38756 @hi-rustin
  - Fix the issue that the LEADING hint does not take effect in UNION ALL statements #50067 @hawkingrei
  - Fix the issue that BIT type columns might cause query errors due to decode failures when they are involved in calculations of some functions #49566 #50850 #50855 @jiyfhust
  - Fix the issue that TiDB might panic when performing a rolling upgrade using tiup cluster upgrade/start due to an interaction issue with PD #50152 @zimulala
  - Fix the issue that executing UNIQUE index lookup with an ORDER BY clause might cause an error #49920 @jackysp

- Fix the issue that TiDB returns wrong query results when processing ENUM or SET types by constant propagation #49440 @winoros
- Fix the issue that TiDB might panic when a query contains the Apply operator and the fatal error: concurrent map writes error occurs #50347 @SeaRise
- Fix the issue that the control of SET_VAR for variables of the string type might become invalid #50507 @qw4990
- Fix the issue that the SYSDATE() function incorrectly uses the time in the plan cache when tidb_sysdate_is_now is set to 1 #49299 @hawkingrei
- Fix the issue that when executing the CREATE GLOBAL BINDING statement, if the schema name is in uppercase, the binding does not take effect #50646 @qw4990
- Fix the issue that Index Path selects duplicated indexes #50496 @AilinKid
- Fix the issue that PLAN REPLAYER fails to load bindings when the CREATE GLOBAL BINDING statement contains IN() #43192 @King-Dylan
- Fix the issue that when multiple analyze tasks fail, the failure reasons are not recorded correctly #50481 @hi-rustin
- Fix the issue that tidb_stats_load_sync_wait does not take effect #50872 @jiyfhust
- Fix the issue that max_execute_time settings at multiple levels interfere with each other #50914 @jiyfhust
- Fix the issue of thread safety caused by concurrent updating of statistics #50835 @hi-rustin
- Fix the issue that executing auto analyze on a partition table might cause TiDB to panic #51187 @hi-rustin
- Fix the issue that SQL bindings might not work when IN() in a SQL statement contains a different number of values #51222 @hawkingrei
- Fix the issue that TiDB cannot correctly convert the type of a system variable in an expression #43527 @hi-rustin
- Fix the issue that TiDB does not listen to the corresponding port when force-init-stats is configured #51473 @hawkingrei
- Fix the issue that in determinate mode (tidb_opt_objective='determinate'), if a query does not contain

predicates, statistics might not be loaded #48257 @time-and-fate

- Fix the issue that the init-stats process might cause TiDB to panic and the load stats process to quit #51581 @hawkingrei
- Fix the issue that the query result is incorrect when the IN() predicate contains NULL #51560 @winoros
- Fix the issue that blocked DDL statements are not displayed in the MDL View when a DDL task involves multiple tables #47743 @wjhuang2016
- Fix the issue that the processed_rows of the ANALYZE task on a table might exceed the total number of rows in that table #50632 @hawkingrei
- Fix the goroutine leak issue that might occur when the HashJoin operator fails to spill to disk #50841 @wshwsh12
- Fix the goroutine leak issue that occurs when the memory usage of CTE queries exceed limits #50337 @guo-shaoge
- Fix the Can't find column ... error that might occur when aggregate functions are used for group calculations #50926 @qw4990
- Fix the issue that DDL operations such as renaming tables are stuck when the CREATE TABLE statement contains specific partitions or constraints #50972 @lcwangchao
- Fix the issue that the monitoring metric tidb_statistics_auto_analyze_total on Grafana is not displayed as an integer #51051 @hawkingrei
- Fix the issue that the tidb_gogc_tuner_threshold system variable is not adjusted accordingly after the tidb_server_memory_limit variable is modified #48180 @hawkingrei
- Fix the issue that the index out of range error might occur when a query involves JOIN operations #42588 @AilinKid
- Fix the issue that getting the default value of a column returns an error if the column default value is dropped #50043 #51324 @crazycs520
- Fix the issue that wrong results might be returned when TiFlash late materialization processes associated columns #49241 #51204 @Lloyd-Pottiger
- Fix the issue that the LIKE() function might return wrong results when processing binary collation inputs #50393 @yibin87

- Fix the issue that the JSON_LENGTH() function returns wrong results when the second parameter is NULL #50931 @SeaRise
- Fix the issue that CAST(AS DATETIME) might lose time precision under certain circumstances #49555 @SeaRise
- Fix the issue that parallel Apply might generate incorrect results when the table has a clustered index #51372 @guo-shaoge
- Fix the issue that ALTER TABLE ... COMPACT TIFLASH REPLICA might incorrectly end when the primary key type is VARCHAR #51810 @breezewish
- Fix the issue that the check on the NULL value of the DEFAULT NULL attribute is incorrect when exchanging partitioned tables using the EXCHANGE PARTITION statement #47167 @jiyfhust
- Fix the issue that the partition table definition might cause wrong behavior when using a non-UTF8 character set #49251 @YangKeao
- Fix the issue that incorrect default values are displayed in the INFORMATION_SCHEMA.VARIABLES_INFO table for some system variables #49461 @jiyfhust
- Fix the issue that no error is reported when empty strings are used as database names in some cases #45873 @yoshikipom
- Fix the issue that the SPLIT TABLE ... INDEX statement might cause TiDB to panic #50177 @Defined2014
- Fix the issue that querying a partitioned table of KeyPartition type might cause an error #50206 #51313 #51196 @time-and-fate @jiyfhust @mjonss
- Fix the issue that querying a Hash partitioned table might produce incorrect results #50427 @Defined2014
- Fix the issue that opentracing does not work correctly #50508 @Defined2014
- Fix the issue that the error message is not complete when ALTER INSTANCE RELOAD TLS reports an error #50699 @dveeden
- Fix the issue that the AUTO_INCREMENT attribute causes non-consecutive IDs due to unnecessary transaction conflicts when assigning auto-increment IDs #50819 @tiancaiamao
- Fix the issue of incomplete stack information in TiDB logs for some errors #50849 @tiancaiamao

    - Fix the issue of excessive memory usage in some queries when the number in the LIMIT clause is too large #51188 @Defined2014
    - Fix the issue that the TTL feature causes data hotspots due to incorrect data range splitting in some cases #51527 @lcwangchao
    - Fix the issue that the SET statement does not take effect when it appears on the first line of an explicit transaction #51387 @YangKeao
    - Fix the issue that querying JSON of BINARY type might cause an error in some cases #51547 @YangKeao
    - Fix the issue that TTL does not handle the transition for daylight saving time adjustments correctly when calculating expiration times #51675 @lcwangchao
    - Fix the issue that the SURVIVAL_PREFERENCES attribute might not appear in the output of the SHOW CREATE PLACEMENT POLICY statement under certain conditions #51699 @lcwangchao
    - Fix the issue that the configuration file does not take effect when it contains an invalid configuration item #51399 @Defined2014
- TiKV
    - Fix the issue that enabling tidb_enable_row_level_checksum might cause TiKV to panic #16371 @cfzjywxk
    - Fix the issue that hibernated Regions are not promptly awakened in exceptional circumstances #16368 @LykxSassinator
    - Fix the issue that the entire Region becomes unavailable when one replica is offline, by checking the last heartbeat time of all replicas of the Region before taking a node offline #16465 @tonyxuqqi
    - Fix the issue that JSON integers greater than the maximum INT64 value but less than the maximum UINT64 value are parsed as FLOAT64 by TiKV, resulting in inconsistency with TiDB #16512 @YangKeao
    - Fix the issue that the monitoring metric tikv_unified_read_pool_thread_count has no data in some cases #16629 @YuJuncen

- PD
  - Fix the issue that data race occurs when the MergeLabels function is called #7535 @lhy1024
  - Fix the issue that there is no output when the evict-leader-scheduler interface is called #7672 @CabinfeverB
  - Fix the issue that the PD monitoring item learner-peer-count does not synchronize the old value after a leader switch #7728 @CabinfeverB
  - Fix the memory leak issue that occurs when watch etcd is not turned off correctly #7807 @rleungx
  - Fix the issue that some TSO logs do not print the error cause #7496 @CabinfeverB
  - Fix the issue that there are unexpected negative monitoring metrics after restart #4489 @lhy1024
  - Fix the issue that the Leader lease expires later than the log time #7700 @CabinfeverB
  - Fix the issue that TiDB panics when TLS switches between TiDB (the PD client) and PD are inconsistent #7900 #7902 #7916 @CabinfeverB
  - Fix the issue that Goroutine leaks when it is not closed properly #7782 @HuSharp
  - Fix the issue that pd-ctl cannot remove a scheduler that contains special characters #7798 @JmPotato
  - Fix the issue that the PD client might be blocked when obtaining TSO #7864 @CabinfeverB
- TiFlash
  - Fix the issue that TiFlash might panic due to unstable network connections with PD during replica migration #8323 @JaySon-Huang
  - Fix the issue that the memory usage increases significantly due to slow queries #8564 @JinheLin
  - Fix the issue that removing and then re-adding TiFlash replicas might lead to data corruption in TiFlash #8695 @JaySon-Huang
  - Fix the issue that TiFlash replica data might be accidentally deleted after performing point-in-time recovery (PITR) or

executing FLASHBACK CLUSTER TO, which might result in data anomalies #8777 @JaySon-Huang
    - Fix the issue that TiFlash panics after executing ALTER TABLE ... MODIFY COLUMN ... NOT NULL, which changes nullable columns to non-nullable #8419 @JaySon-Huang
    - Fix the issue that in the disaggregated storage and compute architecture, queries might be permanently blocked after network isolation #8806 @JinheLin
    - Fix the issue that in the disaggregated storage and compute architecture, TiFlash might panic during shutdown #8837 @JaySon-Huang
    - Fix the issue that TiFlash might crash due to data race in case of remote reads #8685 @solotzg
    - Fix the issue that the CAST(AS JSON) function does not de-duplicate the JSON object key #8712 @SeaRise
    - Fix the issue that the ENUM column might cause TiFlash to crash during chunk encoding #8674 @yibin87
- Tools
    - Backup & Restore (BR)
        - Fix the issue that the log backup checkpoint gets stuck when a Region is split or merged immediately after it becomes a leader #16469 @YuJuncen
        - Fix the issue that TiKV panics when a full backup fails to find a peer in some extreme cases #16394 @Leavrth
        - Fix the issue that log backup gets stuck after changing the TiKV IP address on the same node #50445 @3pointer
        - Fix the issue that BR cannot retry when encountering an error while reading file content from S3 #49942 @Leavrth
        - Fix the issue that when resuming from a checkpoint after data restore fails, an error the target cluster is not fresh occurs #50232 @Leavrth
        - Fix the issue that stopping a log backup task causes TiDB to crash #50839 @YuJuncen
        - Fix the issue that data restore is slowed down due to absence of a leader on a TiKV node #50566 @Leavrth

        - Fix the issue that full restore still requires the target cluster to be empty after the --filter option is specified #51009 @3pointer
    - TiCDC
        - Fix the issue that the file sequence number generated by the storage service might not increment correctly when using the storage sink #10352 @CharlesCheung96
        - Fix the issue that TiCDC returns the ErrChangeFeedAlreadyExists error when concurrently creating multiple changefeeds #10430 @CharlesCheung96
        - Fix the issue that after filtering out add table partition events is configured in ignore-event, TiCDC does not replicate other types of DML changes for related partitions to the downstream #10524 @CharlesCheung96
        - Fix the issue that the changefeed reports an error after TRUNCATE PARTITION is executed on the upstream table #10522 @sdojjy
        - Fix the issue that snapshot lost caused by GC is not reported in time when resuming a changefeed and the checkpoint-ts of the changefeed is smaller than the GC safepoint of TiDB #10463 @sdojjy
        - Fix the issue that TiCDC fails to validate TIMESTAMP type checksum due to time zone mismatch after data integrity validation for single-row data is enabled #10573 @3AceShowHand
        - Fix the issue that the Syncpoint table might be incorrectly replicated #10576 @asddongmen
        - Fix the issue that OAuth2.0, TLS, and mTLS cannot be enabled properly when using Apache Pulsar as the downstream #10602 @asddongmen
        - Fix the issue that a changefeed might get stuck when TiKV upgrades, restarts, or evicts a leader #10584 @asddongmen
        - Fix the issue that data is written to a wrong CSV file due to wrong BarrierTS in scenarios where DDL statements are executed frequently #10668 @lidezhu

        - Fix the issue that data race in the KV client causes TiCDC to panic #10718 @asddongmen
        - Fix the issue TiCDC panics when scheduling table replication tasks #10613 @CharlesCheung96
    - TiDB Data Migration (DM)
        - Fix the issue that data is lost when the upstream primary key is of binary type #10672 @GMHDBJD
    - TiDB Lightning
        - Fix the performance regression issue caused by checking TiKV space #43636 @lance6716
        - Fix the issue that TiDB Lightning reports an error when encountering invalid symbolic link files during file scanning #49423 @lance6716
        - Fix the issue that TiDB Lightning fails to correctly parse date values containing `0` when `NO_ZERO_IN_DATE` is not included in `sql_mode` #50757 @GMHDBJD

## 2.6 Contributors

We would like to thank the following contributors from the TiDB community:

- Aoang
- bufferflies
- daemon365
- eltociear
- lichunzhu
- jiyfhust
- pingandb
- shenqidebaozi
- Smityz
- songzhibin97
- tangjingyu97
- Tema
- ub-3
- yoshikipom

# 3 TiDB 7.6.0 Release Notes

Release date: January 25, 2024

TiDB version: 7.6.0

Quick access: Quick start

7.6.0 introduces the following key features and improvements:

| Category | Feature/Enhancement | Description |
|---|---|---|
| Scalability and Performance | Cross-database SQL binding | When managing hundreds of databases with the same schema, it is often necessary to apply SQL bindings across these databases. For example, in SaaS or PaaS data platforms, each user typically operates separate databases with the same schema and runs similar SQL queries on them. In this case, it is impractical to bind SQL for each database one by one. TiDB v7.6.0 introduces cross-database SQL bindings that enable matching bindings across all schema-equivalent databases. |
| | Achieve up to 10 times faster for snapshot restore (experimental) | BR v7.6.0 introduces an experimental coarse-grained Region scatter algorithm to accelerate snapshot restores for clusters. In clusters with many TiKV nodes, this algorithm significantly improves cluster resource efficiency by more evenly distributing load across nodes and better utilizing per-node network bandwidth. In several real-world cases, this improvement accelerates restore process by about up to 10 times. |
| | Achieve up to 10 times faster for creating tables in batch (experimental) | With the implementation of the new DDL architecture in v7.6.0, the performance of batch table creation has witnessed a remarkable improvement, up to 10 times faster. This substantial enhancement drastically reduces the time needed for creating numerous tables. This acceleration is particularly noteworthy in SaaS scenarios, where the prevalence of high volumes of tables, ranging from tens to hundreds of thousands, is a common challenge. |
| | Use Active PD Followers to enhance PD's Region information query service (experimental) | TiDB v7.6.0 introduces an experimental feature "Active PD Follower", which allows PD followers to provide Region information query services. This feature improves the capability of the PD cluster to handle GetRegion and ScanRegions requests in |

| Category | Feature/Enhancement | Description |
|---|---|---|
| | | clusters with a large number of TiDB nodes and Regions, thereby reducing the CPU pressure on the PD leader. |
| Reliability and Availability | Support TiProxy (experimental) | Full support for the TiProxy service, easily deployable via deployment tooling, to manage and maintain connections to TiDB so that they live through rolling restarts, upgrades, or scaling events. |
| | Data Migration (DM) officially supports MySQL 8.0 (GA) | Previously, using DM to migrate data from MySQL 8.0 is an experimental feature and is not available for production environments. TiDB v7.6.0 enhances the stability and compatibility of this feature to help you smoothly and quickly migrate data from MySQL 8.0 to TiDB in production environments. In v7.6.0, this feature becomes generally available (GA). |

## 3.1 Feature details

### 3.1.1 Scalability

- Use the Active PD Follower feature to enhance the scalability of PD's Region information query service (experimental) #7431 @CabinfeverB

  In a TiDB cluster with a large number of Regions, the PD leader might experience high CPU load due to the increased overhead of handling heartbeats and scheduling tasks. If the cluster has many TiDB instances, and there is a high concurrency of requests for Region information, the CPU pressure on the PD leader increases further and might cause PD services to become unavailable.

  To ensure high availability, TiDB v7.6.0 supports using the Active PD Follower feature to enhance the scalability of PD's Region information query service. You can enable the Active PD Follower feature by setting the system variable pd_enable_follower_handle_region to ON. After this feature is enabled, TiDB evenly distributes Region information requests to all PD servers, and PD followers can also handle Region requests, thereby reducing the CPU pressure on the PD leader.

  For more information, see documentation.

### 3.1.2 Performance

- BR improves snapshot restore speed by up to 10 times (experimental) #33937 #49886 @3pointer

  As a TiDB cluster scales up, it becomes increasingly crucial to quickly restore the cluster from failures to minimize business downtime. Before v7.6.0, the Region scattering algorithm is a primary bottleneck in performance restoration. In v7.6.0, BR optimizes the Region scattering algorithm, which quickly splits the restore task into a large number of small tasks and scatters them to all TiKV nodes in batches. The new parallel recovery algorithm fully utilizes the resources of each TiKV node, thereby achieving a rapid parallel recovery. In several real-world cases, the snapshot restore speed of the cluster is improved by about 10 times in large-scale Region scenarios.

  The new coarse-grained Region scatter algorithm is experimental. To use it, you can configure the `--granularity="coarse-grained"` parameter in the `br` command. For example:

  ```
  br restore full \
  --pd "${PDIP}:2379" \
  --storage "s3://${Bucket}/${Folder}" \
  --s3.region "${region}" \
  --granularity "coarse-grained" \
  --send-credentials-to-tikv=true \
  --log-file restorefull.log
  ```

  For more information, see documentation.

- The Titan engine is enabled by default #16245 @Connor1996 @v01dstar @tonyxuqqi

  To better support TiDB wide table write scenarios, especially with support for JSON, starting from TiDB v7.6.0, the Titan engine is enabled by default. The Titan engine automatically segregates large values exceeding 32 KB from RocksDB’s LSM Tree, and stores them separately in Titan to optimize the handling of large values. The Titan engine is completely compatible with RocksDB features utilized by TiKV. This strategic shift not only diminishes write amplification effect, but also enhances performance in write, update, and point-query scenarios involving large values. Additionally, in Range Scan scenarios, the Titan engine’s optimization has resulted in performance comparable to that of RocksDB in the default configuration.

This configuration change remains compatible with earlier versions. For existing TiDB clusters, when upgrading to TiDB v7.6.0 or a later version, the Titan engine is disabled by default. You have the flexibility to manually enable or disable the Titan engine based on your specific requirements.

For more information, see documentation.

- Support pushing down the following string functions to TiKV #48170 @gengliqi

  - LOWER()
  - UPPER()

  For more information, see documentation.

- Support pushing down the following JSON functions to TiFlash #48350 #48986 #48994 #49345 #49392 @SeaRise @yibin87

  - JSON_UNQUOTE()
  - JSON_ARRAY()
  - JSON_DEPTH()
  - JSON_VALID()
  - JSON_KEYS()
  - JSON_CONTAINS_PATH()

  For more information, see documentation.

- Improve the performance of creating tables by 10 times (experimental) #49752 @gmhdbjd

  In previous versions, when migrating tens of thousands of tables from the upstream database to TiDB, it is time-consuming and inefficient for TiDB to create these tables. Starting from v7.6.0, TiDB introduces a new TiDB DDL V2 architecture. You can enable it by configuring the system variable tidb_ddl_version. Compared with previous versions, the new version of the DDL improves the performance of creating batch tables by 10 times, and significantly reduces time for creating tables.

  For more information, see documentation.

- Support periodic full compaction (experimental) #12729 afeinberg

  Starting from v7.6.0, TiDB supports periodic full compaction for TiKV. This feature serves as an enhancement to Garbage Collection (GC) to eliminate redundant data versions. In scenarios where application activity shows obvious peaks and valleys, you can use this feature to perform data compaction during idle periods to improve the performance during peak periods.

  You can set the specific times that TiKV initiates periodic full compaction by configuring the TiKV configuration item periodic-full-compact-start-times and limit the maximum CPU usage rate for TiKV periodic full compaction by configuring periodic-full-compact-start-max-cpu. The default value of periodic-full-compact-start-max-cpu is 10%, which means that periodic full compaction is triggered only when the CPU utilization of TiKV is lower than 10%, thereby reducing the impact on application traffic.

  For more information, see documentation.

### 3.1.3 Reliability

- Cross-database execution plan binding #48875 @qw4990

  When running SaaS services on TiDB, it is common practice to store data for each tenant in separate databases for easier data maintenance and management. This results in hundreds of databases with the same table and index definitions, and similar SQL statements. In such scenario, when you create an execution plan binding for a SQL statement, this binding usually applies to the SQL statements in other databases as well.

  For this scenario, TiDB v7.6.0 introduces the cross-database binding feature, which supports binding the same execution plan to SQL statements with the same schema, even if they are in different databases. When creating a cross-database binding, you need to use the wildcard * to represent the database name, as shown in the following example. After the binding is created, regardless of which database the tables t1 and t2 are in, TiDB will try to use this binding to generate an execution plan for any SQL statement with the same schema, which saves your effort to create a binding for each database.

```
CREATE GLOBAL BINDING FOR
USING
  SELECT /*+ merge_join(t1, t2) */ t1.id, t2.amount
  FROM *.t1, *.t2
  WHERE t1.id = t2.id;
```

In addition, cross-database binding can effectively mitigate SQL performance issues caused by uneven distribution and rapid changes in user data and workload. SaaS providers can use cross-database binding to fix execution plans validated by users with large data volumes, thereby fixing execution plans for all users. For SaaS providers, this feature provides significant convenience and experience improvements.

Due to the system overhead (less than 1%) introduced by cross-database binding, TiDB disables this feature by default. To use cross-database binding, you need to first enable the tidb_opt_enable_fuzzy_binding system variable.

For more information, see documentation.

### 3.1.4 Availability

- Support the proxy component TiProxy (experimental) #413 @djshow832 @xhebox

  TiProxy is the official proxy component of TiDB, located between the client and TiDB server. It provides load balancing and connection persistence functions for TiDB, making the workload of the TiDB cluster more balanced and not affecting user access to the database during maintenance operations.

  - During maintenance operations such as rolling restarts, rolling upgrades, and scaling-in in a TiDB cluster, changes occur in the TiDB servers which result in interruptions in connections between clients and the TiDB servers. By using TiProxy, connections can be smoothly migrated to other TiDB servers during these maintenance operations so that clients are not affected.
  - Client connections to a TiDB server cannot be dynamically migrated to other TiDB servers. When the workload of multiple TiDB servers is unbalanced, it might result in a situation where

the overall cluster resources are sufficient, but certain TiDB servers experience resource exhaustion leading to a significant increase in latency. To address this issue, TiProxy provides dynamic migration for connection, which allows connections to be migrated from one TiDB server to another without any impact on the clients, thereby achieving load balancing for the TiDB cluster.

TiProxy has been integrated into TiUP, TiDB Operator, and TiDB Dashboard, making it easy to configure, deploy and maintain.

For more information, see documentation.

### 3.1.5 SQL

- LOAD DATA supports explicit transactions and rollbacks #49079 @ekexium

  Compared with MySQL, the transactional behavior of the LOAD DATA statement varies in different TiDB versions before v7.6.0, so you might need to make additional adjustments when using this statement. Specifically, before v4.0.0, LOAD DATA commits every 20000 rows. From v4.0.0 to v6.6.0, TiDB commits all rows in one transaction by default and also supports committing every fixed number of rows by setting the tidb_dml_batch_size system variable. Starting from v7.0.0, tidb_dml_batch_size no longer takes effect on LOAD DATA and TiDB commits all rows in one transaction.

  Starting from v7.6.0, TiDB processes LOAD DATA in transactions in the same way as other DML statements, especially in the same way as MySQL. The LOAD DATA statement in a transaction no longer automatically commits the current transaction or starts a new transaction. Moreover, you can explicitly commit or roll back the LOAD DATA statement in a transaction. Additionally, the LOAD DATA statement is affected by the TiDB transaction mode setting (optimistic or pessimistic transaction). These improvements simplify the migration process from MySQL to TiDB and offer a more unified and controllable experience for data import.

  For more information, see documentation.

### 3.1.6 DB operations

- FLASHBACK CLUSTER supports specifying a precise TSO #48372 @BornChanger

  In TiDB v7.6.0, the flashback feature is more powerful and precise. It not only supports rolling back a cluster to a specified historical timestamp but also enables you to specify a precise recovery TSO using FLASHBACK CLUSTER TO TSO, thereby increasing flexibility in data recovery. For example, you can use this feature with TiCDC. After pausing data replication and conducting pre-online read-write tests in your downstream TiDB cluster, this feature allows the cluster to gracefully and quickly roll back to the paused TSO and continue to replicate data using TiCDC. This streamlines the pre-online validation process and simplifies data management.

  ```
  FLASHBACK CLUSTER TO TSO 445494839813079041;
  ```

  For more information, see documentation.

- Support automatically terminating long-running idle transactions #48714 @crazycs520

  In scenarios where network disconnection or application failure occurs, COMMIT/ROLLBACK statements might fail to be transmitted to the database. This could lead to delayed release of database locks, causing transaction lock waits and a rapid increase in database connections. Such issues are common in test environments but can also occur occasionally in production environments, and they are sometimes difficult to diagnose promptly. To avoid these issues, TiDB v7.6.0 introduces the tidb_idle_transaction_timeout system variable, which automatically terminates long-running idle transactions. When a user session in a transactional state remains idle for a duration exceeding the value of this variable, TiDB will terminate the database connection of the transaction and roll back it.

  For more information, see documentation.

- Simplify the syntax for creating execution plan bindings #48876 @qw4990

  TiDB v7.6.0 simplifies the syntax for creating execution plan bindings. When creating an execution plan binding, you no longer need to provide the original SQL statement. TiDB identifies the original SQL statement based on the statement with hints. This improvement enhances the convenience of creating execution plan bindings. For example:

```sql
CREATE GLOBAL BINDING
USING
SELECT /*+ merge_join(t1, t2) */ * FROM t1, t2 WHERE t1.id = t2.id;
```

For more information, see documentation.

- Support dynamically modifying the size limit of a single row record in TiDB #49237 @zyguan

  Before v7.6.0, the size of a single row record in a transaction is limited by the TiDB configuration item txn-entry-size-limit. If the size limit is exceeded, TiDB returns the entry too large error. In this case, you need to manually modify the TiDB configuration file and restart TiDB to make the modification take effect. To reduce your management overhead, TiDB v7.6.0 introduces the system variable tidb_txn_entry_size_limit, which supports dynamically modifying the value of the txn-entry-size-limit configuration item. The default value of this variable is 0, which means that TiDB uses the value of the configuration item txn-entry-size-limit by default. When this variable is set to a non-zero value, TiDB limits the size of a row record in transactions to the value of this variable. This improvement enhances the flexibility for you to adjust system configurations without restarting TiDB.

  For more information, see documentation.

- BR restores system tables by default, such as user data #48567 @BornChanger #49627 @Leavrth

  Starting from v5.1.0, when you back up snapshots, BR automatically backs up system tables in the mysql schema, but does not restore these system tables by default. In v6.2.0, BR adds the parameter --with-sys-table to support restoring data in some system tables, providing more flexibility in operations.

  To further reduce your management overhead and provide more intuitive default behavior, starting from v7.6.0, BR enables the parameter --with-sys-table by default and supports restoring user data for the cloud_admin user. This means that BR restores some system tables by default during restoration, especially user account and table statistics data. This improvement makes backup and restore operations more intuitive, thereby reducing the burden of manual configuration and improving the overall operation experience.

For more information, see documentation.

### 3.1.7 Observability

- Enhance observability related to resource control #49318 @glorv @bufferflies @nolouch

  As more and more users use resource groups to isolate application workloads, Resource Control provides enhanced data based on resource groups. This helps you monitor resource group workloads and settings, ensuring that you can quickly identify and accurately diagnose problems, including:

  - Slow Queries: add the resource group name, resource unit (RU) consumption, and time for waiting for resources.
  - Statement Summary Tables: add the resource group name, RU consumption, and time for waiting for resources.
  - In the system variable tidb_last_query_info, add a new entry ru_consumption to indicate the consumed RU by SQL statements. You can use this variable to get the resource consumption of the last statement in the session.
  - Add database metrics based on resource groups: QPS/TPS, execution time (P999/P99/P95), number of failures, and number of connections.
  - Add the system table request_unit_by_group to record the history records of daily consumed RUs of all resource groups.

  For more information, see Identify Slow Queries, Statement Summary Tables, and Key Monitoring Metrics of Resource Control.

### 3.1.8 Data migration

- Data Migration (DM) support for migrating MySQL 8.0 becomes generally available (GA) #10405 @lyzx2001

  Previously, using DM to migrate data from MySQL 8.0 is an experimental feature and is not available for production environments. TiDB v7.6.0 enhances the stability and compatibility of this feature to help you smoothly and quickly migrate data from MySQL 8.0 to TiDB in production environments. In v7.6.0, this feature becomes generally available (GA).

For more information, see documentation.

- TiCDC supports replicating DDL statements in bi-directional replication (BDR) mode (experimental) #10301 #48519 @okJiang @asddongmen

  Starting from v7.6.0, TiCDC supports replication of DDL statements with bi-directional replication configured. Previously, replicating DDL statements was not supported by TiCDC, so users of TiCDC's bi-directional replication had to apply DDL statements to both TiDB clusters separately. With this feature, TiCDC allows for a cluster to be assigned the `PRIMARY` BDR role, and enables the replication of DDL statements from that cluster to the downstream cluster.

  For more information, see documentation.

- TiCDC supports querying the downstream synchronization status of a changefeed #10289 @hongyunyan

  Starting from v7.6.0, TiCDC introduces a new API `GET /api/v2/changefeed/{changefeed_id}/synced` to query the downstream synchronization status of a specified replication task (changefeed). By using this API, you can determine whether the upstream data received by TiCDC has been synchronized to the downstream system completely.

  For more information, see documentation.

- TiCDC adds support for three-character delimiters with CSV output protocol #9969 @zhangjinpeng87

  Starting from v7.6.0, you can specify the CSV output protocol delimiters as 1 to 3 characters long. With this change, you can configure TiCDC to generate file output using two-character delimiters (such as `||` or `$^`) or three-character delimiters (such as `|@|`) to separate fields in the output.

  For more information, see documentation.

## 3.2 Compatibility changes

**Note:**

This section provides compatibility changes you need to know when you upgrade from v7.5.0 to the current version (v7.6.0). If you are upgrading from v7.4.0 or earlier versions to the current version, you might also need to check the compatibility changes introduced in intermediate versions.

### 3.2.1 MySQL compatibility

- Before TiDB v7.6.0, the LOAD DATA operation commits all rows in a single transaction or commits transactions in a batch, which is slightly different from MySQL behaviors. Starting from v7.6.0, TiDB processes LOAD DATA in transactions in the same way as MySQL. The LOAD DATA statement in a transaction no longer automatically commits the current transaction or starts a new transaction. Moreover, you can explicitly commit or roll back the LOAD DATA statement in a transaction. Additionally, the LOAD DATA statement is affected by the TiDB transaction mode setting (optimistic or pessimistic transaction). #49079 @ekexium

### 3.2.2 System variables

| Variable name | Change type | Description |
|---|---|---|
| tidb_auto_analyze_partition_batch_size | Modified | Changes the default value from 1 to 128 after further tests. |
| tidb_sysproc_scan_concurrency | Modified | In a large-scale cluster, the concurrency of scan operations can be adjusted higher to meet the needs of ANALYZE. Therefore, change the maximum value from 256 to 4294967295. |
| tidb_analyze_distsql_scan_concurrency | Newly added | Sets the concurrency of the scan operation when executing the ANALYZE operation. The default value is 4. |
| tidb_ddl_version | Newly added | Controls whether to enable TiDB DDL V2. Set the value to 2 to enable it and 1 to disable it. The default value is 1. When TiDB DDL V2 is enabled, DDL statements will be executed using TiDB DDL V2. The execution speed of DDL statements for creating tables is increased by 10 times compared with TiDB DDL V1. |
| tidb_enable_global_index | Newly added | Controls whether to support creating Global indexes for partitioned tables. The default value is OFF. Global |

| Variable name | Change type | Description |
|---|---|---|
| | | index is currently in the development stage. **It is not recommended to modify the value of this system variable**. |
| tidb_idle_transaction_timeout | Newly added | Controls the idle timeout for transactions in a user session. When a user session is in a transactional state and remains idle for a duration exceeding the value of this variable, TiDB will terminate the session. The default value 0 means unlimited. |
| tidb_opt_enable_fuzzy_binding | Newly added | Controls whether to enable the cross-database binding feature. The default value OFF means cross-database binding is disabled. |
| tidb_txn_entry_size_limit | Newly added | Dynamically modifies the TiDB configuration item performance.txn-entry-size-limit. It limits the size of a single row of data in TiDB. The default value of this variable is 0, which means that TiDB uses the value of the configuration item txn-entry-size-limit by default. When this variable is set to a non-zero value, txn-entry-size-limit is also set to the same value. |
| pd_enable_follower_handle_region | Newly added | Controls whether to enable the Active PD Follower feature (experimental). When the value is OFF, TiDB only obtains Region information from the PD leader. When the value is ON, TiDB evenly distributes requests for Region information to all PD servers, and PD followers can also handle Region requests, thereby reducing the CPU pressure on the PD leader. |

### 3.2.3 Configuration file parameters

| Configuration file | Configuration parameter | Change type | Description |
|---|---|---|---|
| TiDB | tls-version | Modified | The default value is "". The default supported TLS versions of TiDB are changed from TLS1.1 or higher to TLS1.2 or higher. |
| TiKV | blob-file-compression | Modified | The algorithm used for compressing values in Titan, which takes value as the unit. Starting from TiDB v7.6.0, the default compression algorithm is zstd. |
| TiKV | rocksdb.defaultcf.titan.min-blob-size | Modified | Starting from TiDB v7.6.0, the default value for new clusters is 32KB. For existing clusters upgrading to v7.6.0, the default value 1KB remains unchanged. |

| Configuration file | Configuration parameter | Change type | Description |
|---|---|---|---|
| TiKV | rocksdb.titan.enabled | Modified | Enables or disables Titan. For v7.5.0 and earlier versions, the default value is false. Starting from v7.6.0, the default value is true for only new clusters. Existing clusters upgraded to v7.6.0 or later versions will retain the original configuration. |
| TiKV | gc.num-threads | Newly added | When enable-compaction-filter is set to false, this parameter controls the number of GC threads. The default value is 1. |
| TiKV | raftstore.periodic-full-compact-start-times | Newly added | Sets the specific times that TiKV initiates periodic full compaction. The default value [] means periodic full compaction is disabled. |
| TiKV | raftstore.periodic-full-compact-start-max-cpu | Newly added | Limits the maximum CPU usage rate for TiKV periodic full compaction. The default value is 0.1. |
| TiKV | zstd-dict-size | Newly added | Specifies the zstd dictionary compression size. The default value is "0KB", which means to disable the zstd dictionary compression. |
| TiFlash | logger.level | Modified | Changes the default value from "debug" to "INFO" to reduce the cost of logging. |
| TiDB Lightning | tidb.pd-addr | Modified | Configures the addresses of the PD Servers. Starting from v7.6.0, TiDB supports setting multiple PD addresses. |
| TiDB Lightning | block-size | Newly added | Controls the I/O block size for sorting local files in Physical Import Mode (backend='local'). The default value is 16KiB. When the disk IOPS is a bottleneck, you can increase this value to improve performance. |
| BR | --granularity | Newly added | Uses the coarse-grained Region scatter algorithm (experimental) by specifying --granularity="coarse-grained". This |

| Configuration file | Configuration parameter | Change type | Description |
|---|---|---|---|
| | | | accelerates restore speed in large-scale Region scenarios. |
| TiCDC | compression | Newly added | Controls the behavior to compress redo log files. |
| TiCDC | sink.cloud-storage-config | Newly added | Sets the automatic cleanup of historical data when replicating data to object storage. |

### 3.2.4 System tables

- Add a new system table INFORMATION_SCHEMA.KEYWORDS to display the information of all keywords supported by TiDB.
- In the system table INFORMATION_SCHEMA.SLOW_QUERY, add the following fields related to Resource Control:
  - Resource_group: the resource group that the statement is bound to.
  - Request_unit_read: the total read RUs consumed by the statement.
  - Request_unit_write: the total write RUs consumed by the statement.
  - Time_queued_by_rc: the total time that the statement waits for available resources.

## 3.3 Offline package changes

Starting from v7.6.0, the TiDB-community-server binary-package now includes tiproxy-{version}-linux-{arch}.tar.gz, which is the installation package for the proxy component TiProxy.

## 3.4 Deprecated features

- Support for the TLSv1.0 and TLSv1.1 protocols is deprecated in TiDB v7.6.0 and will be removed in v8.0.0. Please upgrade to TLSv1.2 or TLSv1.3.
- The baseline evolution feature for execution plans will be deprecated in TiDB v8.0.0. The equivalent functionality will be redesigned in the subsequent versions.
- The tidb_disable_txn_auto_retry system variable will be deprecated in TiDB v8.0.0. After that, TiDB will no longer support automatic retries of optimistic transactions.

## 3.5 Improvements

- TiDB
  - When a non-binary collation is set and the query includes LIKE, the optimizer generates an IndexRangeScan to improve the execution efficiency #48181 #49138 @time-and-fate
  - Enhance the ability to convert OUTER JOIN to INNER JOIN in specific scenarios #49616 @qw4990
  - Improve the balance of Distributed eXecution Framework (DXF) tasks in the scenario where nodes are restarted #47298 @ywqzzy
  - Support multiple accelerated ADD INDEX DDL tasks to be queued for execution, instead of falling back to normal ADD INDEX tasks #47758 @tangenta
  - Improve the compatibility of ALTER TABLE ... ROW_FORMAT #48754 @hawkingrei
  - Modify the CANCEL IMPORT JOB statement to a synchronous statement #48736 @D3Hunter
  - Improve the speed of adding indexes to empty tables #49682 @zimulala
  - When the columns of a correlated subquery are not referenced by the upper-level operator, the correlated subquery can be eliminated directly #45822 @King-Dylan
  - EXCHANGE PARTITION operations now trigger maintenance updates of statistics #47354 @hi-rustin
  - TiDB supports building binary files that meet the requirements of Federal Information Processing Standards (FIPS) #47948 @tiancaiamao
  - Optimize the TiDB implementation when handling some type conversions and fix related issues #47945 #47864 #47829 #47816 @YangKeao @lcwangchao
  - When obtaining the schema version, TiDB uses the KV timeout feature to read by default, reducing the impact of slow meta Region leader reads on schema version updates #48125 @cfzjywxk
- TiKV

  - Add an API endpoint /async_tasks for querying asynchronous tasks #15759 @YuJuncen
  - Add priority labels to gRPC monitoring to display resource group data of different priorities #49318 @bufferflies
  - Support dynamically adjusting the value of readpool.unified.max-tasks-per-worker, which can calculate the number of running tasks separately based on priority #16026 @glorv
  - Support dynamically adjusting the number of GC threads, with a default value of 1 #16101 @tonyxuqqi
- PD
  - Improve the availability of PD TSO during disk jitter #7377 @HuSharp
- TiFlash
  - Reduce the impact of disk performance jitter on read latency #8583 @JaySon-Huang
  - Reduce the impact of background GC tasks on read and write task latency #8650 @JaySon-Huang
  - Support merging identical data reading operations in a storage-compute separation architecture to improve data scanning performance under high concurrency #6834 @JinheLin
  - Optimize the execution performance of SEMI JOIN and LEFT OUTER SEMIJOIN when only JOIN KEY equality conditions are included in JOIN ON #47424 @gengliqi
- Tools
  - Backup & Restore (BR)
    - Support authentication using Amazon S3 session-token and assume-role during full backup recovery phase #39832 @3pointer
    - Introduce a new integration test for Point-In-Time Recovery (PITR) in the delete range scenario, enhancing PITR stability #47738 @Leavrth
    - Improve the table creation performance of the RESTORE statement in scenarios with large datasets #48301 @Leavrth

- Refactor the BR exception handling mechanism to increase tolerance for unknown errors #47656 @3pointer

- TiCDC

  - Improve the performance of TiCDC replicating data to object storage by increasing parallelism #10098 @CharlesCheung96
  - Support making TiCDC Canal-JSON content format compatible with the content format of the official Canal output by setting content-compatible=true in the sink-uri configuration #10106 @3AceShowHand

- TiDB Data Migration (DM)

  - Add configurations for full data physical import to DM OpenAPI #10193 @GMHDBJD

- TiDB Lightning

  - Support configuring multiple PD addresses to enhance stability #49515 @mittalrishabh
  - Support configuring the block-size parameter to control the I/O block size for sorting local files to improve performance #45037 @mittalrishabh

## 3.6 Bug fixes

- TiDB

  - Fix the issue that TiDB panics and reports an error invalid memory address or nil pointer dereference #42739 @CbcWestwolf
  - Fix the TiDB node panic issue that occurs when DDL jobID is restored to 0 #46296 @jiyfhust
  - Fix the issue that the same query plan has different PLAN_DIGEST values in some cases #47634 @King-Dylan
  - Fix the issue that executing UNION ALL with the DUAL table as the first subnode might cause an error #48755 @winoros
  - Fix the issue that queries containing common table expressions (CTEs) report runtime error: index out of range [32] with length 32 when tidb_max_chunk_size is set to a small value #48808 @guo-shaoge
  - Fix the issue of Goroutine leak when using AUTO_ID_CACHE=1 #46324 @tiancaiamao

- Fix the issue that the result of COUNT(INT) calculated by MPP might be incorrect #48643 @AilinKid
- Fix the issue that executing ALTER TABLE ... LAST PARTITION fails when the partition column type is DATETIME #48814 @crazycs520
- Fix the issue that using the _ wildcard in LIKE when the data contains trailing spaces can result in incorrect query results #48983 @time-and-fate
- Fix the issue that high CPU usage of TiDB occurs due to long-term memory pressure caused by tidb_server_memory_limit #48741 @XuHuaiyu
- Fix the issue that the query result is incorrect when an ENUM type column is used as the join key #48991 @winoros
- Fix the issue that queries containing CTEs unexpectedly get stuck when the memory limit is exceeded #49096 @AilinKid
- Fix the issue that TiDB server might consume a significant amount of resources when the enterprise plugin for audit logging is used #49273 @lcwangchao
- Fix the issue that the optimizer incorrectly converts TiFlash selection path to the DUAL table in specific scenarios #49285 @AilinKid
- Fix the issue that UPDATE or DELETE statements containing WITH RECURSIVE CTEs might produce incorrect results #48969 @winoros
- Fix the issue that a query containing the IndexHashJoin operator gets stuck when memory exceeds tidb_mem_quota_query #49033 @XuHuaiyu
- Fix the issue that in non-strict mode (sql_mode = ''), truncation during executing INSERT still reports an error #49369 @tiancaiamao
- Fix the issue that CTE queries might report an error type assertion for CTEStorageMap failed during the retry process #46522 @tiancaiamao
- Fix the issue that LIMIT and ORDER BY might be invalid in nested UNION queries #49377 @AilinKid
- Fix the issue that parsing invalid values of ENUM or SET types would directly cause SQL statement errors #49487 @winoros

- Fix the issue of excessive statistical error in constructing statistics caused by Golang's implicit conversion algorithm #49801 @qw4990
- Fix the issue that Daylight Saving Time is displayed incorrectly in some time zones #49586 @overvenus
- Fix the issue that tables with AUTO_ID_CACHE=1 might lead to gRPC client leaks when there are a large number of tables #48869 @tiancaiamao
- Fix the issue that TiDB server might panic during graceful shutdown #36793 @bb7133
- Fix the issue that ADMIN RECOVER INDEX reports ERROR 1105 when processing a table containing CommonHandle #47687 @Defined2014
- Fix the issue that specifying placement rules when executing ALTER TABLE t PARTITION BY reports the error ERROR 8239 #48630 @mjonss
- Fix the issue that the START_TIME column type in INFORMATION_SCHEMA.CLUSTER_INFO is not valid #45221 @dveeden
- Fix the issue that invalid EXTRA column type in INFORMATION_SCHEMA.COLUMNS leads to the error Data Too Long, field len 30, data len 45 #42030 @tangenta
- Fix the issue that IN (...) causes different plan digests in INFORMATION_SCHEMA.STATEMENTS_SUMMARY #33559 @King-Dylan
- Fix the issue that when converting the TIME type to the YEAR type, the returned result mixes TIME and the year #48557 @YangKeao
- Fix the issue that disabling tidb_enable_collect_execution_info causes the coprocessor cache to panic #48212 @you06
- Fix the issue that TiDB crashes when shuffleExec quits unexpectedly #48230 @wshwsh12
- Fix the issue that static CALIBRATE RESOURCE relies on the Prometheus data #49174 @glorv
- Fix the issue that when adding a large interval to a date, it returns an incorrect result. After the fix, an interval with an invalid prefix or the string true is treated as zero, which is consistent with MySQL 8.0 #49227 @lcwangchao

- Fix the issue that the ROW function incorrectly infers the null type and causes an unexpected error #49015 @wshwsh12
- Fix the issue that the ILIKE function might cause data race in some scenarios #49677 @lcwangchao
- Fix the issue that query results are incorrect due to STREAM_AGG() incorrectly handling CI #49902 @wshwsh12
- Fix the issue that encoding fails when converting bytes to TIME #47346 @wshwsh12
- Fix the issue that the behavior of the ENFORCED option in the CHECK constraint is inconsistent with MySQL 8.0 #47567 #47631 @jiyfhust
- Fix the issue that DDL statements with the CHECK constraint are stuck #47632 @jiyfhust
- Fix the issue that adding index fails for DDL statements due to out of memory #47862 @GMHDBJD
- Fix the issue that upgrading the cluster during executing ADD INDEX might cause the data to be inconsistent with the indexes #46306 @zimulala
- Fix the issue that executing ADMIN CHECK after updating the tidb_mem_quota_query system variable returns ERROR 8175 #49258 @tangenta
- Fix the issue that when ALTER TABLE modifies the type of a column referenced by a foreign key, the change in DECIMAL precision is not reported as an error #49836 @yoshikipom
- Fix the issue that when ALTER TABLE modifies the type of a column referenced by a foreign key, the change in INTEGER length is reported as an error by mistake #47702 @yoshikipom
- Fix the issue that in some scenarios the expression index does not detect that the divisor is 0 #50053 @lcwangchao
- Mitigate the issue that TiDB nodes might encounter OOM errors when dealing with a large number of tables #50077 @zimulala
- Fix the issue that DDL gets stuck in the running state during cluster rolling restart #50073 @tangenta
- Fix the issue that results might be incorrect when accessing global indexes of partitioned tables using PointGet or BatchPointGet operators #47539 @L-maple

- Fix the issue that MPP plans might not be selected when indexes on generated columns are set as visible #47766 @AilinKid
- Fix the issue that LIMIT might not be pushed down to the OR type Index Merge #48588 @AilinKid
- Fix the issue that duplicate built-in rows might exist in the mysql.bind_info table after BR import #46527 @qw4990
- Fix the issue that statistics for partitioned tables are not updated as expected after partitions are dropped #48182 @hi-rustin
- Fix the issue that errors might be returned during the concurrent merging of global statistics for partitioned tables #48713 @hawkingrei
- Fix the issue that query results might be incorrect when the LIKE operator is used for index range scans on a column with PADDING SPACE #48821 @time-and-fate
- Fix the issue that generated columns might trigger concurrent read and write on memory and result in data race #44919 @tangenta
- Fix the issue that ANALYZE TABLE might still collect Top1 statistics even when WITH 0 TOPN (indicating not collecting topN statistics) is specified #49080 @hawkingrei
- Fix the issue that illegal optimizer hints might cause valid hints to be ineffective #49308 @hawkingrei
- Fix the issue that statistics for Hash partitioned tables are not correspondingly updated when you add, drop, reorganize, or TRUNCATE partitions #48235 #48233 #48226 #48231 @hi-rustin
- Fix the issue that after the time window for automatic statistics updates is configured, statistics might still be updated outside that time window #49552 @hawkingrei
- Fix the issue that old statistics are not automatically deleted when a partitioned table is converted to a non-partitioned table #49547 @hi-rustin
- Fix the issue that old statistics are not automatically deleted when you clear data from a non-partitioned table using TRUNCATE TABLE #49663 @hi-rustin
- Fix the issue that enforced sorting might become ineffective when a query uses optimizer hints (such as STREAM_AGG()) that

enforce sorting and its execution plan contains IndexMerge #49605 @AilinKid

- Fix the issue that histogram statistics might not be parsed into readable strings when the histogram boundary contains NULL #49823 @AilinKid
- Fix the issue that executing queries containing the GROUP_CONCAT(ORDER BY) syntax might return errors #49986 @AilinKid
- Fix the issue that UPDATE, DELETE, and INSERT statements return overflow errors instead of warnings when the SQL_MODE is not strict #49137 @YangKeao
- Fix the issue that data cannot be inserted when a table has a composite index consisting of multi-valued indexes and non-binary type strings #49680 @YangKeao
- Fix the issue that LIMIT in multi-level nested UNION queries might become ineffective #49874 @Defined2014
- Fix the issue that querying partitioned tables with the BETWEEN ... AND ... condition returns incorrect results #49842 @Defined2014
- Fix the issue that hints cannot be used in REPLACE INTO statements #34325 @YangKeao
- Fix the issue that TiDB might select the wrong partition when querying Hash partitioned tables #50044 @Defined2014
- Fix the connection error that occurs when you use MariaDB Connector/J with compression enabled #49845 @onlyacat

• TiKV

- Fix the issue that the damaged SST files might spread to other TiKV nodes and cause TiKV to panic #15986 @Connor1996
- Fix the issue that Online Unsafe Recovery cannot handle merge abort #15580 @v01dstar
- Fix the issue that the joint state of DR Auto-Sync might time out when scaling out #15817 @Connor1996
- Fix the issue that blob-run-mode in Titan cannot be updated online #15978 @tonyxuqqi
- Fix the issue that Resolved TS might be blocked for two hours #11847 #15520 #39130 @overvenus

- Fix the issue that Flashback might get stuck when encountering notLeader or regionNotFound #15712 @HuSharp
    - Fix the issue that if TiKV runs extremely slowly, it might panic after Region merge #16111 @overvenus
    - Fix the issue that TiKV cannot read in-memory pessimistic locks when GC scans expired locks #15066 @cfzjywxk
    - Fix the issue that the blob file size in Titan monitoring is incorrect #15971 @Connor1996
    - Fix the issue that replicating large tables using TiCDC might cause TiKV to OOM #16035 @overvenus
    - Fix the issue that TiDB and TiKV might produce inconsistent results when processing DECIMAL arithmetic multiplication truncation #16268 @solotzg
    - Fix the issue that cast_duration_as_time might return incorrect results #16211 @gengliqi
    - Fix the issue that TiKV converts the time zone incorrectly for Brazil and Egypt #16220 @overvenus
    - Fix the issue that TiKV might panic when gRPC threads are checking is_shutdown #16236 @pingyu
- PD

    - Fix the issue that the etcd health check in PD does not remove expired addresses #7226 @iosmanthus
    - Fix the issue that when PD leader is transferred and there is a network partition between the new leader and the PD client, the PD client fails to update the information of the leader #7416 @CabinfeverB
    - Fix some security issues by upgrading the version of Gin Web Framework from v1.8.1 to v1.9.1 #7438 @niubell
    - Fix the issue that the orphan peer is deleted when the number of replicas does not meet the requirements #7584 @bufferflies
- TiFlash

    - Fix the issue of memory leak when TiFlash encounters memory limitation during query #8447 @JinheLin
    - Fix the issue that data of TiFlash replicas would still be garbage collected after executing FLASHBACK DATABASE #8450 @JaySon-Huang

- Fix the issue that the memory usage increases significantly due to slow queries #8564 @JinheLin
- Fix the issue that some TiFlash replica data cannot be recovered through RECOVER TABLE or FLASHBACK TABLE in scenarios with frequent execution of CREATE TABLE and DROP TABLE #1664 @JaySon-Huang
- Fix the issue that query results are incorrect when querying with filtering conditions like ColumnRef in (Literal, Func...) #8631 @Lloyd-Pottiger
- Fix the TiFlash panic issue when TiFlash encounters conflicts during concurrent DDL execution #8578 @JaySon-Huang
- Fix the issue that TiFlash might not be able to select the GC owner of object storage data under the disaggregated storage and compute architecture #8519 @JaySon-Huang
- Fix the issue that the lowerUTF8 and upperUTF8 functions do not allow characters in different cases to occupy different bytes #8484 @gengliqi
- Fix the issue that TiFlash incorrectly handles ENUM when the ENUM value is 0 #8311 @solotzg
- Fix the incompatibility issue in the INET_NTOA() expression #8211 @solotzg
- Fix the potential OOM issue that might occur when scanning multiple partitioned tables during stream read #8505 @gengliqi
- Fix the issue that short queries executed successfully print excessive info logs #8592 @windtalker
- Fix the issue that TiFlash might crash when it is stopped #8550 @guo-shaoge
- Fix the random invalid memory access issue that might occur with GREATEST or LEAST functions containing constant string parameters #8604 @windtalker

- Tools
  - Backup & Restore (BR)
    - Fix the issue that BR generates incorrect URIs for external storage files #48452 @3AceShowHand

        - Fix the issue that the log backup task can start but does not work properly if failing to connect to PD during task initialization #16056 @YuJuncen
        - Fix the issue that the log backup task might encounter memory leak and fail to run properly after startup #16070 @YuJuncen
        - Fix the issue that inserting data into the system table mysql.gc_delete_range during the PITR process returns an error #49346 @Leavrth
        - Fix the issue that the Unsupported collation error is reported when you restore data from backups of an old version #49466 @3pointer
        - Fix the issue that permissions are not updated timely after user tables are recovered through snapshots in certain scenarios #49394 @Leavrth
    - TiCDC
        - Fix the issue that the WHERE clause does not use the primary key as a condition when replicating DELETE statements in certain scenarios #9812 @asddongmen
        - Fix the issue that the TiCDC server might panic when replicating data to an object storage service #10137 @sdojjy
        - Fix the potential data race issue during kv-client initialization #10095 @3AceShowHand
        - Fix the issue that TiCDC mistakenly closes the connection with TiKV in certain special scenarios #10239 @hicqu
        - Fix the issue that TiCDC server might panic when executing lossy DDL statements in upstream #9739 @hicqu
        - Fix the issue that checkpoint-ts might get stuck when TiCDC replicates data to downstream MySQL #10334 @zhangjinpeng87
    - TiDB Data Migration (DM)
        - Fix the issue that DM encounters “event type truncate not valid” error that causes the upgrade to fail #10282 @GMHDBJD

- Fix the performance degradation issue when replicating data in GTID mode #9676 @feran-morgan-pingcap
- Fix the issue that a migration task error occurs when the downstream table structure contains shard_row_id_bits #10308 @GMHDBJD

### 3.7 Contributors

We would like to thank the following contributors from the TiDB community:

- 0o001 (First-time contributor)
- bagechengzi (First-time contributor)
- feran-morgan-pingcap (First-time contributor)
- highpon
- jiyfhust
- L-maple
- lkshminarayanan (First-time contributor)
- lyang24 (First-time contributor)
- mittalrishabh
- morgo
- nkg- (First-time contributor)
- onlyacat
- shawn0915
- Smityz
- szpnygo (First-time contributor)
- ub-3 (First-time contributor)
- xiaoyawei (First-time contributor)
- yorkhellen
- yoshikipom (First-time contributor)
- Zheaoli