MySQL NDB Cluster 8.4 Release Notes

Abstract

This document contains release notes for the changes in each release of MySQL NDB Cluster that uses version 8.4 of the `NDB` (`NDBCLUSTER`) storage engine and the preceding Innovation series (8.1, 8.2, and 8.3)

Each NDB Cluster 8.4 release is based on a mainline MySQL Server release and a particular version of the `NDB` storage engine, as shown in the version string returned by executing `SELECT VERSION()` in the `mysql` client, or by executing the `ndb_mgm` client `SHOW` or `STATUS` command; for more information, see MySQL NDB Cluster 8.4.

For general information about features added in NDB Cluster 8.4, see What is New in MySQL NDB Cluster 8.4. For a complete list of all bug fixes and feature changes in MySQL NDB Cluster, please refer to the changelog section for each individual NDB Cluster release.

For additional MySQL 8.4 documentation, see the MySQL 8.4 Reference Manual, which includes an overview of features added in MySQL 8.4 that are not specific to NDB Cluster (What Is New in MySQL 8.4 since MySQL 8.0), and discussion of upgrade issues that you may encounter for upgrades from MySQL 8.3 to MySQL 8.4 (Changes in MySQL 8.4). For a complete list of all bug fixes and feature changes made in MySQL 8.4 that are not specific to `NDB`, see MySQL 8.4 Release Notes.

Updates to these notes occur as new product features are added, so that everybody can follow the development process. If a recent version is listed here that you cannot find on the download page (https://dev.mysql.com/downloads/), the version has not yet been released.

The documentation included in source and binary distributions may not be fully up to date with respect to release note entries because integration of the documentation occurs at release build time. For the most up-to-date release notes, please refer to the online documentation instead.

For help with using MySQL, please visit the MySQL Forums, where you can discuss your issues with other MySQL users.

Document generated on: 2026-04-08 (revision: 31055)

Table of Contents

Preface and Legal Notices 2
Changes in MySQL NDB Cluster 8.4.9 (Not yet released) 3
Changes in MySQL NDB Cluster 8.4.8 (2026-01-23) 4
Changes in MySQL NDB Cluster 8.4.7 (2025-10-24) 5
Changes in MySQL NDB Cluster 8.4.6 (2025-07-23) 6
Changes in MySQL NDB Cluster 8.4.5 (2025-04-16) 7
Changes in MySQL NDB Cluster 8.4.4 (2025-01-22) 10
Changes in MySQL NDB Cluster 8.4.3 (2024-10-16) 13
Changes in MySQL NDB Cluster 8.4.2 (2024-07-23) 14
Changes in MySQL NDB Cluster 8.4.1 (2024-07-02) 15
Changes in MySQL NDB Cluster 8.4.0 (2024-04-30) 18
Release Series Changelogs: MySQL NDB Cluster 8.4 22
- Changes in MySQL NDB Cluster 8.4.8 (2026-01-23) 22
- Changes in MySQL NDB Cluster 8.4.7 (2025-10-24) 23
- Changes in MySQL NDB Cluster 8.4.6 (2025-07-23) 24
- Changes in MySQL NDB Cluster 8.4.5 (2025-04-16) 25
- Changes in MySQL NDB Cluster 8.4.4 (2025-01-22) 27
- Changes in MySQL NDB Cluster 8.4.3 (2024-10-16) 29
- Changes in MySQL NDB Cluster 8.4.1 (2024-07-02) 31
- Changes in MySQL NDB Cluster 8.4.0 (2024-04-30) 33

Changes in MySQL NDB Cluster 8.3.0 (2024-01-16) 36
Changes in MySQL NDB Cluster 8.2.0 (2023-10-25) 41
Changes in MySQL NDB Cluster 8.1.0 (2023-07-18) 43
Index 46

Preface and Legal Notices

This document contains release notes for the changes in each release of MySQL NDB Cluster that uses version 8.4 of the `NDB` storage engine.

Legal Notices

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Changes in MySQL NDB Cluster 8.4.9 (Not yet released)

MySQL NDB Cluster 8.4.9 is a new LTS release of NDB 8.4, based on MySQL Server 8.4 and including features in version 8.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 8.4. NDB Cluster 8.4 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of major changes made in NDB Cluster 8.4, see What is New in MySQL NDB Cluster 8.4.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 8.4 through MySQL 8.4.9 (see Changes in MySQL 8.4.9 (Not yet released)).

Version 8.4.9-ndb-8.4.9 has no release notes, or they have not been published because the product version has not been released.

Changes in MySQL NDB Cluster 8.4.8 (2026-01-23)

MySQL NDB Cluster 8.4.8 is a new LTS release of NDB 8.4, based on MySQL Server 8.4 and including features in version 8.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 8.4. NDB Cluster 8.4 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of major changes made in NDB Cluster 8.4, see What is New in MySQL NDB Cluster 8.4.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 8.4 through MySQL 8.4.8 (see Changes in MySQL 8.4.8 (2026-01-20)).

Note

These release notes were created with the assistance of MySQL HeatWave GenAI.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Reporting of File Close errors has been improved. (Bug #38726643)
- The option `--skip-fk-checks` was added to `ndb_restore`. This option modifies the behavior of `--rebuild-indexes` so that, when foreign keys are re-enabled, the existing data in the table is not checked for consistency. (Bug #38593666)
- The logs generated for backup and restore operations are improved in this release. (Bug #38592288)

Bugs Fixed

- The NDB purge process responsible for removing rows from `mysql.ndb_binlog_index` when a binary log file is purged, assumed that `autocommit` is enabled, and each DML operation commits immediately. However, if `autocommit` is disabled globally, the purge process inherited this value and purge operations were not committed.

  As of this release, `autocommit` is enabled for the purge session, even when the global setting is disabled. (Bug #38488185)
- NDB data nodes failed to start if either of the `NDBCNTR` or `DBDIH` system files were empty. (Bug #38424959)
- A data node failure occurred due to a reference validity check which failed. (Bug #38422448)
- BackupRecords in a participant node were not released from the pool when a backup was aborted, causing subsequent backup operations to fail. (Bug #38151265)
- Certain Transporter error messages did not include information on the source node. (Bug #37922543)
- The `ndb_restore` log message for total log bytes restored was incorrect (Bug #37697971)
- In the NDB management server, the TLS INFO statistics `upgraded` and `tls` were incremented if TLS authentication had failed. They should only be incremented on success. (Bug #36414579)

- `ndb_log_bin` could be set dynamically, but had no effect. This variable is not dynamic and is only checked when the binlog thread starts. (Bug #32860560)

Changes in MySQL NDB Cluster 8.4.7 (2025-10-24)

MySQL NDB Cluster 8.4.7 is a new LTS release of NDB 8.4, based on MySQL Server 8.4 and including features in version 8.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 8.4. NDB Cluster 8.4 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of major changes made in NDB Cluster 8.4, see What is New in MySQL NDB Cluster 8.4.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 8.4 through MySQL 8.4.7 (see Changes in MySQL 8.4.7 (2025-10-21)).

Note

These release notes were created with the assistance of MySQL HeatWave GenAI.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Data node logging is improved with the following:
  - Local logs for data node join and leave events.
  - API disconnection handling logs.

  (Bug #38414245)
- MySQL NDB Cluster now generates log entries for redo logging issues, including `redo log buffer exhaustion`, `redo log space exhaustion`, and `file change problems`, which can cause transactions to be aborted, queued, or delayed. A secondary overload control mechanism monitors the rate at which the redo log part's Redo buffer is flushed to disk and estimates the time required to complete writing all data in the part's Redo buffer. (Bug #37903091)
- `LogLevelBackup` and `LogLevelSchema` Data node options were added in this release. (Bug #28004136)

Bugs Fixed

- If a data node restarted during a backup, the remaining data nodes could hang and the backup process stopped. The backup was marked as failed, and other nodes did not terminate their backup process, preventing further backups from being initiated. (Bug #38316252)
- It was not possible to restore backups with `BackupID` values greater than 2147483647, with the `ndb_restore` tool. Errors were returned similar to the following:

  ```
  ndb_restore: [Warning] option 'backupid': signed value
  3000000000 adjusted to 2147483647. Failed to find backup
  2147483647.
  ```

  (Bug #38260769)

- When running CREATE or INSERT statements on a NDB Binlog server with sql_log_bin = 0, replication would fail due to a table not existing. (Bug #37953883)

- Concurrent execution of `TRUNCATE TABLE` statements and starting an Ndb backup could cause the statements to hang indefinitely, affecting cluster availability. Errors were returned similar to the following:

```
ERROR 1296 (HY000): Got error 761 'Unable to drop table as
backup is in progress' from NDBCLUSTER
```

  (Bug #37486661)

- Setting log level options for data nodes in MySQL NDB Cluster, such as `LogLevelCheckpoint`, `LogLevelStatistic`, `LogLevelInfo`, `LogLevelCongestion`, `LogLevelError`, and `LogLevelConnection` to a higher value does not provide additional log information beyond their default values. Errors were returned similar to the following:

```
Ndb kernel thread 0 is stuck in: Job Handling elapsed=100,
Watchdog: Warning overslept
```

  (Bug #17847063)

- The MySQL NDB Cluster's example code for `MGMAPI_LOGEVENT` was intended to compile as plain C, but had a .cpp extension and included a C++ header. The example file has been renamed to `main.c` and the C++ header has been removed. (Bug #12678874)

Changes in MySQL NDB Cluster 8.4.6 (2025-07-23)

MySQL NDB Cluster 8.4.6 is a new LTS release of NDB 8.4, based on MySQL Server 8.4 and including features in version 8.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 8.4. NDB Cluster 8.4 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of major changes made in NDB Cluster 8.4, see What is New in MySQL NDB Cluster 8.4.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 8.4 through MySQL 8.4.6 (see Changes in MySQL 8.4.6 (2025-07-22)).

Note

These release notes were created with the assistance of MySQL HeatWave GenAI.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Timestamps in `NDB` node logs can now be printed with microsecond resolution. Data nodes can enable this feature using the data node `--ndb-log-timestamps=UTC` option; management nodes can also do so using the `ndb_mgmd` option `--ndb-log-timestamps=UTC`. For backwards compatible behavior, you can set this option explicitly to `LEGACY`, which uses the system time zone and resolution in seconds, as in previous releases. In NDB 8.4, this is the default.

For SQL nodes, use the `--log-timestamps` option; be aware that this `mysqld` option does not support `LEGACY` as a value.

See NDB Cluster Log Messages, for more information. (Bug #37924338)

Bugs Fixed

- **NDB Client Programs:** `ndb_restore`, when applying the log, allowed an infinite number of retries due to temporary errors, rather than limiting these to 11 as expected. (Bug #37883579)

  References: This issue is a regression of: Bug #31546136.

- **NDB Client Programs:** When restoring from backup with `DefaultOperationRedoProblemAction=ABORT`, an error in data node handling of a redo log part overload condition resulted in an incorrect error code (626 in this case) being sent back to `ndb_restore`, which caused `ndb_restore` to exit prematurely since it did not expect such an error. (Bug #37687485)

  References: This issue is a regression of: Bug #13219, Bug #13930, Bug #13980.

- Improved password handling for file system encryption.

  Our thanks to Axel Svensson for the contribution. (Bug #37909595)

- `CREATE TABLESPACE`, when the specified logfile group did not exist, was rejected with error `723 No such table existed`, which did not correctly identify the source of the problem with the SQL statement. Now in such cases, the server issues Error `789 Logfile group not found`. (Bug #37802388)

- Warning 1296 `The temporary named table ... already exists` showed the table and schema names in the wrong order.

  Our thanks to Axel Svensson for the contribution. (Bug #117918, Bug #37807409)

Changes in MySQL NDB Cluster 8.4.5 (2025-04-16)

MySQL NDB Cluster 8.4.5 is a new LTS release of NDB 8.4, based on MySQL Server 8.4 and including features in version 8.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 8.4. NDB Cluster 8.4 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of major changes made in NDB Cluster 8.4, see What is New in MySQL NDB Cluster 8.4.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 8.4 through MySQL 8.4.5 (see Changes in MySQL 8.4.5 (2025-04-15)).

- Compilation Notes
- Functionality Added or Changed
- Bugs Fixed

Compilation Notes

- The ability to build the source without NDB using the internal script `storage/ndb/compile-cluster` was adversely affected by work done in NDB 8.0.31 making the `ndbcluster` plugin part of its default build. (Bug #117215, Bug #37484376)

Functionality Added or Changed

- Added the `Ndb_schema_participant_count` status variable. This variable provides the count of MySQL servers which are currently participating in NDB Cluster schema change distribution. (Bug #37529202)

Bugs Fixed

- **NDB Replication:** Timestamps written to the binary log included fractional parts (microseconds) although the query being logged did not use any high-precision functionality. This problem was caused by not resetting the state indicating that fractional parts had been used.

  We fix this by ensuring that the indicator for the use of microseconds in a given query after having run it is always reset. This avoids the possibility of a later query writing a timestamp which includes microseconds to the binary log when the query as executed did not use microseconds. (Bug #37112446)

- **ndbinfo Information Database:** Certain queries against `ndbinfo` tables were not handled correctly. (Bug #37372650)

- **NDB Client Programs:** With a data node in an unstarted state, such as immediately after executing *`node_id`* `RESTART -n` in the `ndb_mgm` client, issuing `ALL REPORT BACKUPSTATUS` in the client subsequently led to an unplanned shutdown of the cluster. (Bug #37505513)

- **MySQL NDB ClusterJ:** The ClusterJ log file only reported the configured, requested node ID for a cluster connection (which was often zero). With this fix, after a connection has been established, ClusterJ reports the actual assigned node ID in the log. (Bug #37556172)

- **MySQL NDB ClusterJ:** A potential circular reference from `NdbRecordSmartValueHandlerImpl` that can cause delays in garbage collection has been removed. (Bug #37361267)

- **MySQL NDB ClusterJ:** Setting the connection property `com.mysql.clusterj.byte.buffer.pool.sizes` to "`512, 51200`" caused ClusterJ application to fail with a fatal exception thrown by `java.nio.ByteBuffer`. (Bug #37188154)

- **MySQL NDB ClusterJ:** When using a debug build of ClusterJ to run any tests in the testsuite, it exited with the error "1 thread(s) did not exit." (Bug #36383937)

- **MySQL NDB ClusterJ:** Running a ClusterJ application with Java 10 resulted in `java.lang.ClassNotFoundException`, because the class `java.internal.ref.Cleaner` is not available in Java 10. With this fix, the `java.lang.ref.Cleaner` class is used instead for resource cleanup. (Bug #29931569)

- The bundled `libxml2` library has been upgraded to version 2.9.13. (Bug #37806165)

- API node failure is detected by one or more data nodes; data nodes detecting API node failure inform all other data nodes of the failure, eventually triggering API node failure handling on each data node.

  Each data node handles API node failure independently; once all internal blocks have completed cleanup, the API node failure is considered handled, and, after a timed delay, the `QMGR` block allows the failed API node's node ID to be used for new connections.

  `QMGR` monitors API node failure handling, periodically generating warning logs for API node failure handling that has not completed (approximately every 30 seconds). These logs indicate which blocks have yet to complete failure handling.

  This enhancement improves logging in handling stalls particularly with regard to the `DBTC` block, which must roll back or commit and complete the API node's transactions, and release the associated `COMMIT` and `ACK` markers. In addition, the time to wait for API node failure handling is

now configurable as the `ApiFailureHandlingTimeout` data node configuration parameter; after this number of seconds, handling is escalated to a data node restart. (Bug #37524092)

References: See also: Bug #37469364.

- When a data node hangs during shutdown reasons for this may include: I/O problems on the node, in which case the thread shutting down hangs while operating on error and trace files; or an error in the shutdown logic, where the thread shutting down raises a Unix signal, and causes a deadlock. When such issues occur, users might observe watchdog warnings in the logs, referring to the last signal processed; this could be misleading in cases where there was actually a (different) preceding cause which had triggered the shutdown.

  To help pinpoint the origin of such problems if they occur, we have made the following improvements:

  - Added a new watchdog state `shutting down`. This is set early enough in the error handling process that it causes all watchdog logging of shutdown stalls to attribute the delay to a shutdown delay (correctly) rather than problem in execution.
  - We have also modified the watchdog mechanism to be aware of shutdown states, and use a more direct path—which is less likely to stall—to force the data node process to stop when needed.

  (Bug #37518267)

- When restoring `NDB` tables from backup, it is now possible for `mysqld` to open such tables even if their indexes are not yet available. (Bug #37516858)
- Signal dump code run when handling an unplanned node shutdown sometimes exited unexpectedly when speculatively reading section IDs which might not be present. (Bug #37512526)
- The LQH_TRANSCONF signal printer did not validate its input length correctly, which could lead the node process to exit. (Bug #37512477)
- When restoring stored grants (using `ndb_restore --include-stored-grants`) from an `NDB` backup following an initial restart of the data nodes, the `ndb_sql_metadata` table was neither created nor restored. (Bug #37492169)
- Nothing was written to the cluster log to indicate when `PURGE BINARY LOGS` had finished waiting for the purge to complete. (Bug #37489870)
- `WITH_NDB_TLS_SEARCH_PATH` was not set when compiling NDB Cluster using `WITHOUT_SERVER`. (Bug #37398657)
- Now, when `ndb_metadata_check` is enabled, we synchronize both schema and tables in the same interval. (Bug #37382551)
- This fix addresses the following two issues:

  1. When a resend could not be started due to a gap created by an out of buffer error in the event stream, forwarding event data for the bucket was not initiated. We fix this by ensure that the takeover process has been initiated before exiting the resend code.
  2. An out of buffer error which occurred during an ongoing resend was not handled. In this case, we now interrupt the resend when such an error is raised.

  (Bug #37349305)

- On certain rare occasions, when concurrent calls were made to `release()` and `get()`, instances of `ndb_schema_object` were doubly freed. (Bug #35793818)
- If an out-of-buffer-release (OOBR) process took an excessive amount of time, the reset was performed prematurely, before all buffers were released, thus interfering with concurrent seizing of new pages, beginning new `out_of_buffer` handling, or starting a resend.

We solve this issue by ensuring that resumption of event buffering takes place only after the OOBR process has completed for all buckets. (Bug #20648778)

Changes in MySQL NDB Cluster 8.4.4 (2025-01-22)

MySQL NDB Cluster 8.4.4 is a new LTS release of NDB 8.4, based on MySQL Server 8.4 and including features in version 8.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 8.4. NDB Cluster 8.4 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of major changes made in NDB Cluster 8.4, see What is New in MySQL NDB Cluster 8.4.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 8.4 through MySQL 8.4.4 (see Changes in MySQL 8.4.4 (2025-01-21)).

- Compilation Notes
- Bugs Fixed

Compilation Notes

- **macOS:** A `uint64_t` value used with `%zu` caused a `[-Wformat]` compiler warning on MacOS. (Bug #37174692)
- Removed a warning in `storage/ndb/src/common/util/cstrbuf.cpp`. (Bug #37049014)

Bugs Fixed

- **Microsoft Windows:** Successive iterations of the sequence `ndb_sign_keys --create-key` followed by `ndb_sign_keys --promote` were unsuccessful on Windows. (Bug #36951132)
- **NDB Disk Data:** `mysqld` did not use a disk scan for `NDB` tables with 256 disk columns or more. (Bug #37201922)
- **NDB Replication:** The replication applier normally retries temporary errors occurring while applying transactions. Such retry logic is not performed for transactions containing row events where the `STMT_END_F` flag is missing; instead, the statement is committed in an additional step while applying the subsequent `COMMIT` query event when there are still locked tables. Problems arose when committing this statement, because temporary errors were not handled properly. Replica skip error functionality was also affected in that it attempted to skip only the error that occurred when a transaction was committed a second time.

  The binary log contains an epoch transaction with writes from multiple server IDs on the source. The replica then uses `IGNORE_SERVER_IDS` (<last_server_id_in_binlog>) to cause the `STMT_END_F` to be filtered away, thus committing the statement from the COMMIT query log event on the applier. Holding a lock on one of the rows to be updated by the applier triggered error handling, which caused replication to stop with an error, with no retries being performed.

  We now handle such errors, logging all messages in diagnostics areas (as is already done for row log events) and then retrying the transaction. (Bug #37331118)
- **NDB Replication:** When a MySQL server performing binary logging connects to an NDB Cluster, it checks for existing binary logs; if it finds any, it writes an `Incident` event to a log file of its own so that any downstream replicas can detect the potential for lost events. Problems arose under some circumstances because it was possible for the timestamps of events logged in this file to be out of

order; the `Incident` event was written following other events but had a smaller timestamp than these preceding events. We fix this issue by ensuring that a fresh timestamp is used prior to writing an incident to the binary log on startup rather than one which may have been obtained and held for some time previously. (Bug #37228735)

- **NDB Cluster APIs:** The `Ndb_cluster_connection` destructor calls `g_eventLogger::stopAsync()` in order to release the buffers used by the asynchronous logging mechanism as well as to stop the threads responsible for this logging. When the `g_eventLogger` object was deleted before the `Ndb_cluster_connection` destructor was called, the application terminated after trying to use a method on a null object. This could happen in either of two ways:

  - An API program deleted the logger object before deleting the `Ndb_cluster_connection`.
  - `ndb_end()` was called before the `Ndb_cluster_connection` was deleted.

  We solve this issue by skipping the call to `stopAsync()` in the `Ndb_cluster_connection` destructor when `g_eventLogger` is `NULL`. This fix also adds a warning to inform API users that deleting `g_eventLogger` before calling the `Ndb_cluster_connection` destructor is incorrect usage.

  For more information, see API Initialization and Cleanup. (Bug #37300558)

- **NDB Cluster APIs:** Removed known causes of API node versus data node state misalignments, and improved the handling of state misalignments when detected. In one such case, separate handling of scan errors in the `NDB` kernel and those originating in API programs led to cleanup not being performed after some scans. Handling of `DBTC` and API state alignment errors has been improved by this set of fixes, as well as scan protocol timeout handling in `DBSPJ`; now, when such misalignments in state are detected, the involved API nodes are disconnected rather than the data node detecting it being forced to shut down. (Bug #20430083, Bug #22782511, Bug #23528433, Bug #28505289, Bug #36273474, Bug #36395384, Bug #36838756, Bug #37022773, Bug #37022901, Bug #37023549)

  References: See also: Bug #22782511, Bug #23528433, Bug #36273474, Bug #36395384, Bug #36838756.

- **ndbinfo Information Database:** At table create and drop time, access of `ndbinfo` tables such as `operations_per_fragment` and `memory_per_fragment` sometimes examined data which was not valid.

  To fix this, during scans of these `ndbinfo` tables, we ignore any fragments from tables in transient states at such times due to being created or dropped. (Bug #37140331)

- Work done previously to support opening `NDB` tables with missing indexes was intended to allow the features of the MySQL server to be used to solve problems in cases where indexes cannot be rebuilt due to unmet constraints. With missing indexes, some of the SQL handler functionality is unavailable—for example, the use of indexes to select rows for modification efficiently, or to identify duplicates when processing modifications, or to push joins relying on indexes. This could lead to the unplanned shutdown of an NDB Cluster SQL node.

  In such cases, the server now simply returns an error. (Bug #37299071)

- Recent refactoring of the transporter layer added the reporting of the presence of socket shutdown errors, but not their nature. This led to confusion in the common case where a socket shutdown is requested, but the socket is already closed by the peer. To avoid such confusion, this logging has been removed. (Bug #37243135)

  References: This issue is a regression of: Bug #35750771.

- It was not possible to create an `NDB` table with 256 or more `BLOB` columns when also specifying a reduced inline size, as in the following SQL statement:

```
CREATE TABLE t1 (
```

```
   pk INT PRIMARY KEY,
   b1 BLOB COMMENT 'NDB_COLUMN=BLOB_INLINE_SIZE=100',
   b2 BLOB COMMENT 'NDB_COLUMN=BLOB_INLINE_SIZE=100',
   ...,
   b256 BLOB COMMENT 'NDB_COLUMN=BLOB_INLINE_SIZE=100'
) ENGINE=NDBCLUSTER;
```

(Bug #37201818)

- In some cases, the occurrence of node failures during shutdown led to the cluster becoming unrecoverable without manual intervention.

  We fix this by modifying global checkpoint ID (GCI) information propagation (`CopyGCI` mechanism) to reject propagation of any set of GCI information which does not describe the ability to recover the cluster automatically as part of a system restart. (Bug #37163647)

  References: See also: Bug #37162636.

- In some cases, node failures during an otherwise graceful shutdown could lead to a cluster becoming unrecoverable without manual intervention. This fix modifies the generic GCI info propagation mechanism (`CopyGCI`) to reject propagating any set of GCI information which does not describe the ability to recover a cluster automatically. (Bug #37162636)

- Improved variable names used in `start_resend()`, and enhanced related debug messages to users and developers with additional information. (Bug #37157987)

- In certain cases, a `COPY_FRAGREQ` signal did not honor a fragment scan lock. (Bug #37125935)

- In cases where `NDB` experienced an API protocol timeout when attempting to close a scan operation, it considered the `DBTC` `ApiConnectRecord` involved to be lost for further use, at least until the API disconnected and API failure handling within `DBTC` reclaimed the record.

  This has been improved by having the API send a `TCRELEASEREQ` signal to `DBTC` in such cases, performing API failure handling for a single `ApiConnectRecord` within `DBTC`. (Bug #37023661)

  References: See also: Bug #36273474, Bug #36395384, Bug #37022773, Bug #37022901, Bug #37023549.

- For tables using the `NDB` storage engine, the column comment option `BLOB_INLINE_SIZE` was silently ignored for `TINYBLOB` columns, and (silently) defaulted to the hard-coded 256 byte value regardless of the size provided; this was misleading to users.

  To fix this problem, we now specifically disallow `BLOB_INLINE_SIZE` on `TINYBLOB` columns altogether, and `NDB` now prints a warning saying that the column size is defaulting to 256 bytes. (Bug #36725332)

- Testing revealed that a fix for a previous issue which added a check of the `ApiConnectRecord` failure number against the system's current failure number did not initialize the `ApiConnectRecord` failure number in all cases. (Bug #36155195)

  References: This issue is a regression of: Bug #36028828.

- `ndb_config` did not always handle very long file paths correctly.

  Our thanks to Dirkjan Bussink for the contribution. (Bug #116748, Bug #37310680)

- Errors of unknown provenance were logged while assigning node IDs during cluster synchronization, leading to user doubt and concern. Logging of the data node `QMGR` block and the `ndb_mgmd` process relating to node ID allocation issues has therefore been improved, to supply more and better information about what is being reported in such cases. (Bug #116351, Bug #37189356)

- A multi-range scan sometimes lost its fragment lock for the second and subsequent ranges of the scan. (Bug #111932, Bug #35660890)

Changes in MySQL NDB Cluster 8.4.3 (2024-10-16)

MySQL NDB Cluster 8.4.3 is a new LTS release of NDB 8.4, based on MySQL Server 8.4 and including features in version 8.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 8.4. NDB Cluster 8.4 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of major changes made in NDB Cluster 8.4, see What is New in MySQL NDB Cluster 8.4.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 8.4 through MySQL 8.4.3 (see Changes in MySQL 8.4.3 (2024-10-15)).

- Deprecation and Removal Notes
- Functionality Added or Changed
- Bugs Fixed

Deprecation and Removal Notes

- **NDB Client Programs:** The `ndb_size.pl` utility is now deprecated and is no longer supported. You can expect it to be removed from a future version of the NDB Cluster distribution; for this reason, you should now modify any applications which depend on it accordingly. (WL #16456)
- Use of an `Ndb.cfg` file for setting the connection string for an NDB process was not well documented or supported. With this release, this file is now formally deprecated, and you should expect support for it to be removed in a future release of MySQL Cluster. (WL #15765)

Functionality Added or Changed

- The `ndbcluster` plugin subscribes to all changes that occur in `NDB` and writes them epoch by epoch to the binary log. Each epoch received from `NDB` consists of a large number of changes, all of which are written to the binary log transaction cache before flushing them to the binary log. Previously, it was possible to configure the cache size for all threads, which often led to improper resource allocation for a MySQL Server used for writing a binary log of changes for `NDB`.

  To enable dimensioning and configuring the system properly, we introduce a new system variable `ndb_log_cache_size` which makes it possible to set the size of the transaction cache used by the `NDB` binary log injector, so that this size can be set separately for writing the binary log for NDB transactions and (using `binlog_cache_size`) for writing other transactions whose sizes are likely to be smaller. (Bug #36694848)

Bugs Fixed

- **NDB Cluster APIs:** Using `NdbRecord` and `OO_SETVALUE` from the NDB API to write the value of a `Varchar`, `Varbinary`, `Longvarchar`, or `Longvarbinary` column failed with error `829`. (Bug #36989337)
- **MySQL NDB ClusterJ:** References to ClusterJPA and OpenJPA have been removed from the comments in the packaging files, as JPA code was already removed from ClusterJ some time ago. (Bug #36725675)
- **MySQL NDB ClusterJ:** `ReconnectTest` in the ClusterJ test suite failed sometimes due to a race condition. The test has been rewritten with proper synchronization. (Bug #28550140)
- Removed node management code from `TRIX` that was not actually used. (Bug #37006547)

- Submitting concurrent shutdown commands for individual nodes using `ndb_mgm SHUTDOWN node_id` or the MGM API sometimes had one or both of the following adverse results:

  - Cluster failure when all nodes in the same node group were stopped
  - Inability to recover when all nodes in the same node group were stopped, and the cluster had more than one node group

  This was due to the fact that the (planned) shutdown of a single node assumed that only one such shutdown occurred at a time, but did not actually check this limitation.

  We fix this so that concurrent single-node shutdown requests are serialized across the cluster, and any which would cause a cluster outage are rejected. (Bug #36943756)

  References: See also: Bug #36839995.

- Shutdown of a data node late in a schema transaction updating index statistics caused the president node to shut down as well. (Bug #36886242)

  References: See also: Bug #36877952.

- It was possible for duplicate events to be sent to user applications when a data node was shut down. (Bug #36750146)

- `BLOB_INLINE_SIZE=0` set within a column comment was not honored, and the default for the blob type was used instead (such as 256 bytes for `BLOB`).

  See NDB_COLUMN Options, for more information. (Bug #36724336)

- Issues arose when an attempt was made to use a SHM transporter's wakeup socket before it was ready, due in part to error-handling when setting up the SHM transporter, which did not close the socket correctly prior to making another attempt at setup. (Bug #36568752, Bug #36623058)

- An error in a `my.cnf` file could cause the management node to shut down unexpectedly. (Bug #36508565)

- A race condition sometimes occurred between the watchdog thread and the signal execution thread trying to start node failure handling in parallel. (Bug #35728261)

Changes in MySQL NDB Cluster 8.4.2 (2024-07-23)

MySQL NDB Cluster 8.4.2 is a new LTS release of NDB 8.4, based on MySQL Server 8.4 and including features in version 8.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 8.4. NDB Cluster 8.4 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of major changes made in NDB Cluster 8.4, see What is New in MySQL NDB Cluster 8.4.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 8.4 through MySQL 8.4.2 (see Changes in MySQL 8.4.2 (2024-07-23)).

This release contains no functional changes specific to MySQL NDB Cluster, and is published to align with and include changes made in MySQL Server 8.4.2.

Changes in MySQL NDB Cluster 8.4.1 (2024-07-02)

MySQL NDB Cluster 8.4.1 is a new LTS release of NDB 8.4, based on MySQL Server 8.4 and including features in version 8.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 8.4. NDB Cluster 8.4 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of major changes made in NDB Cluster 8.4, see What is New in MySQL NDB Cluster 8.4.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 8.4 through MySQL 8.4.1 (see Changes in MySQL 8.4.1 (2024-07-01)).

Important

This release is no longer available for download. It was removed due to a critical issue that could stop the server from restarting following the creation of a very large number of tables (8001 or more). Please upgrade to MySQL Cluster 8.4.2 instead.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change:** Now, when the removal of a data node file or directory fails with a file does not exist (**`ENOENT`**) error, this is treated as a successful removal.
- **ndbinfo Information Database:** Added a `type` column to the `transporter_details` table in the `ndbinfo` information database. This column shows the type of connection used by the transporter, which is either of `TCP` or `SHM`.
- **NDB Client Programs:** Added the `--CA-days` option to `ndb_sign_keys` to make it possible to specify a certificate's lifetime. (Bug #36549567)
- **NDB Client Programs:** When started, `ndbd` now produces a warning in the data node log like this one:

  ```
  2024-05-28 13:32:16 [ndbd] WARNING  -- Running ndbd with a single thread of
  signal execution.  For multi-threaded signal execution run the ndbmtd binary.
  ```

  (Bug #36326896)

Bugs Fixed

- **NDB Replication:** When subscribing to changes in the `mysql.ndb_apply_status` table, different settings were used depending on whether `ndb_log_apply_status` was `ON` or `OFF`. Since `ndb_log_apply_status` can be changed at runtime and subscriptions are not recreated at that time, changing these settings at runtime did not have the desired effect.

  The difference between enabling `ndb_log_apply_status` dynamically at runtime and doing so from the start of the MySQL proccess was in the format used when writing the ndb_apply_status updates to the binary log. When `ndb_log_apply_status` was enabled at runtime, writes were still done using the `UPDATE` format when `WRITE` was intended.

  To fix this inconsistency and make the behavior more distinct, we now always use `WRITE` format in such cases; using the `WRITE` format also makes the binary log image slightly smaller and is thus

preferred. In addition, the cleanup of old events has been improved, which improves the cleanup of failed attempts to create tables and events. (Bug #36453684)

- **NDB Replication:** The binary log index purge callback was skipped for the replica applier, which caused orphan rows to be left behind in the `ndb_binlog_index` table. (Bug #20573020, Bug #35847745, Bug #36378551, Bug #36420628, Bug #36423593, Bug #36485220, Bug #36492736)
- **NDB Cluster APIs:** It was possible to employ the following NDB API methods without them being used as `const`, although this alternative usage had long been deprecated (and was not actually documented):

  - `Dictionary::listEvents()`
  - `Dictionary::listIndexes()`
  - `Dictionary::listObjects()`
  - `NdbOperation::getNdbErrorLine()`

  Now, each of these methods must always be invoked as `const`. (Bug #36165876)
- **NDB Client Programs:** `ndb_redo_log_reader` could not read data from encrypted files. (Bug #36313482)
- **NDB Client Programs:** `ndb_redo_log_reader` exited with `Record type = 0 not implemented` when reaching an unused page, all zero bytes, or a page which was only partially used (typically a page consisting of the page header only). (Bug #36313259)
- **NDB Client Programs:** `ndb_restore` did not restore a foreign key whose columns differed in order from those of the parent key.

  Our thanks to Axel Svensson for the contribution. (Bug #114147, Bug #36345882)
- The destructor for `NDB_SCHEMA_OBJECT` makes several assertions about the state of the schema object, but the state was protected by a mutex, and the destructor did not acquire this mutex before testing the state.

  We fix this by acquiring the mutex within the destructor. (Bug #36568964)
- `NDB` now writes a message to the MySQL server log before and after logging an incident in the binary log. (Bug #36548269)
- Removed a memory leak in `/util/NodeCertificate.cpp`. (Bug #36537931)
- Removed a memory leak from `src/ndbapi/NdbDictionaryImpl.cpp`. (Bug #36532102)
- The internal method `CertLifetime::set_set_cert_lifetime(X509 *cert)` should set the not-before and not-after times in the certificate to the same as those stored in the `CertLifetime` object, but instead it set the not-before time to the current time, and the not-after time to be of the same duration as the object. (Bug #36514834)
- Removed a possible use-after-free warning in `ConfigObject::copy_current()`. (Bug #36497108)
- When a thread acquires and releases the global schema lock required for schema changes and reads, the associated log message did not identify who performed the operation.

  To fix this issue, we now do the following:

  - Prepend the message in the log with the identification of the NDB Cluster component or user session responsible.
  - Provide information about the related Performance Schema thread so that it can be traced.

(Bug #36446730)

References: See also: Bug #36446604.

- Metadata changes were not logged with their associated thread IDs. (Bug #36446604)

  References: See also: Bug #36446730.

- When building `NDB` using `lld`, the build terminated prematurely with the error message `ld.lld: error: version script assignment of 'local' to symbol 'my_init' failed: symbol not defined` while attempting to link `libndbclient.so`. (Bug #36431274)

- TLS did not fail cleanly on systems which used OpenSSL 1.0, which is unsupported. Now in such cases, users get a clear error message advising that an upgrade to OpenSSL 1.1 or later is required to use TLS with NDB Cluster. (Bug #36426461)

- The included `libxml2` library was updated to version 2.9.13. (Bug #36417013)

- NDB Cluster's pushdown join functionality expects pushed conditions to filter exactly, so that no rows that do not match the condition must be returned, and all rows that do match the condition must returned. When the condition contained a BINARY value compared to a `BINARY` column this was not always true; if the value was shorter than the column size, it could compare as equal to a column value despite having different lengths, if the condition was pushed down to `NDB`.

  Now, when deciding whether a condition is pushable, we also make sure that the `BINARY` value length exactly matches the `BINARY` column's size. In addition, when binary string values were used in conditions with `BINARY` or `VARBINARY` columns, the actual length of a given string value was not used but rather an overestimate of its length. This is now changed; this should allow more conditions comparing short string values with `VARBINARY` columns to be pushed down than before this fix was made. (Bug #36390313, Bug #36513270)

  References: See also: Bug #36399759, Bug #36400256. This issue is a regression of: Bug #36364619.

- Setting `AutomaticThreadConfig` and `NumCPUs` when running single-threaded data nodes (`ndbd`) sometimes led to unrecoverable errors. Now `ndbd` ignores settings for these parameters, which are intended to apply only to multi-threaded data nodes (`ndbmtd`). (Bug #36388981)

- Improved the error message returned when trying to add a primary key to an `NDBCLUSTER` table using `ALGORITHM=INPLACE`. (Bug #36382071)

  References: See also: Bug #30766579.

- The handling of the `LQH` operation pool which occurs as part of TC takeover skipped the last element in either of the underlying physical pools (static or dynamic). If this element was in use, holding an operation record for a transaction belonging to a transaction coordinator on the failed node, it was not returned, resulting in an incomplete takeover which sometimes left operations behind. Such operations interfered with subsequent transactions and the copying process (`CopyFrag`) used by the failed node to recover.

  To fix this problem, we avoid skipping the final record while iterating through the `LQH` operation records during TC takeover. (Bug #36363119)

- The `libssh` library was updated to version 0.10.4. (Bug #36135621)

- When distribution awareness was not in use, the cluster tended to choose the same data node as the transaction coordinator repeatedly. (Bug #35840020, Bug #36554026)

- In certain cases, management nodes were unable to allocate node IDs to restarted data and SQL nodes. (Bug #35658072)

- Setting `ODirect` in the cluster's configuration caused excess logging when verifying that `ODirect` was actually settable for all paths. (Bug #34754817)

- In some cases, when trying to perform an online add index operation on an `NDB` table with no explicit primary key (see Limitations of NDB online operations), the resulting error message did not make the nature of the problem clear. (Bug #30766579)

  References: See also: Bug #36382071.

Changes in MySQL NDB Cluster 8.4.0 (2024-04-30)

MySQL NDB Cluster 8.4.0 is a new development release of NDB 8.4, based on MySQL Server 8.4 and including features in version 8.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 8.4. NDB Cluster 8.4 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of major changes made in NDB Cluster 8.4, see What is New in MySQL NDB Cluster 8.4.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 8.4 through MySQL 8.4.0 (see Changes in MySQL 8.4.0 (2024-04-30)).

- Deprecation and Removal Notes
- ndbinfo Information Database
- Functionality Added or Changed
- Bugs Fixed

Deprecation and Removal Notes

- **Packaging; Linux:** Removed the deprecated tool `/usr/bin/pathfix.py` from packages for Fedora 39. (Bug #35997178)

- The unused `INFORMATION_SCHEMA.TABLESPACES` table, deprecated in MySQL 8.0.22, has now been removed.

  The Information Schema `FILES` table provides tablespace-related information for `NDB` tables. (WL #14065)

ndbinfo Information Database

- The `ndbinfo transporter_details` table, introduced in NDB 8.0, provides information about individual transporters used in an NDB Cluster, rather than aggregate data as shown by the `transporters` table.

  This release adds the following columns to `transporter_details`:

  - `sendbuffer_used_bytes`: Number of bytes of signal data currently stored pending send using this transporter.
  - `sendbuffer_max_used_bytes`: Historical maximum number of bytes of signal data stored pending send using this transporter. Reset when the transporter connects.
  - `sendbuffer_alloc_bytes`: Number of bytes of send buffer currently allocated to store pending send bytes for this transporter. Send buffer memory is allocated in large blocks which may be sparsely used.

- `sendbuffer_max_alloc_bytes`: Historical maximum number of bytes of send buffer allocated to store pending send bytes for this transporter.

for more information, see The ndbinfo transporter_details Table. . (WL #7662)

Functionality Added or Changed

- **Packaging:** Added support for Fedora 40 and Ubuntu 24.04.
- **NDB Replication:** Previously, when SQL nodes performing binary logging had `log_replica_updates=OFF`, replicated updates applied on a replica NDB cluster were still sent to the SQL nodes performing binary logging. Such updates, as well as any updates that do not trigger logging, are no longer sent, in order to decrease network traffic and resource consumption. (WL #15407)
- **ndbinfo Information Database:** Added the `transporter_details` table to the `ndbinfo` information database. This table is similar to the `transporters` table, but provides information about individual transporters rather than in the aggregate.

  For more information, see The ndbinfo transporter_details Table. (Bug #113163, Bug #36031560)
- **NDB Client Programs:** Added the `--verbose` option to the `ndb_waiter` test program to control the verbosity level of the output. (Bug #34547034)
- Improved logging related to purging of the binary log, including start and completions times, and whether it is the injector which has initiated the purge. (Bug #36176983)

Bugs Fixed

- **NDB Replication:** Replication of an `NDB` table stopped under the following conditions:
  - The table had no explicit primary key
  - The table contained `BIT` columns
  - A hash scan was used to find the rows to be updated or deleted

  To fix this issue, we now make sure that the hash keys for the table match on the source and the replica. (Bug #34199339)
- **NDB Cluster APIs:** TLS connection errors were printed even though TLS was not specified for connections.

  To fix this issue, following an ignored TLS error, we explicitly reset the error condition in the management handle to **`NO_ERROR`**. (Bug #36354973)
- **NDB Cluster APIs:** The `NdbEventOperation` methods `hasError()` and `clearError()`, long deprecated, are effectively disabled: `hasError()` now returns a constant 0, and `clearError()` does nothing. To determine an event type, use `getEventType2()` instead.
- **NDB Client Programs:** In some cases, it was not possible to load cerificates generated using `ndb_sign_keys`. (Bug #36430004)
- **NDB Client Programs:** The following command-line options did not function correctly for the `ndb_redo_log_reader` utility program:
  - `--mbyte`
  - `--page`
  - `--pageindex`

(Bug #36313427)

- **NDB Client Programs:** A certificate lifetime generated by `ndb_sign_keys` should consist of a fixed number of days, plus a random amount of extra time provided by the OpenSSL function `RAND_bytes()`, casting the result to a signed integer value. Because this value could sometimes be negative, this led to extra time being subtracted rather than added.

  We eliminate this problem by using an unsigned integer type to hold the value obtained from `RAND_bytes()`. (Bug #36270629)

- **NDB Client Programs:** Invoking ndb_mgmd with the --bind-address option could in some cases cause the program to terminate unexpectedly. (Bug #36263410)

- **NDB Client Programs:** Some `NDB` utilities such `ndb_show_tables` leaked memory from API connections when TLS was required by the data nodes, and with valid certificates. (Bug #36170703)

- **NDB Client Programs:** Work begun in NDB 8.0.18 and 8.0.20 to remove the unnecessary text `NDBT_ProgramExit ...` from the output of NDB programs is completed in this release. This message should no longer appear in the release binaries of any such programs. (Bug #36169823)

  References: See also: Bug #27096741.

- **NDB Client Programs:** The output from `ndb_waiter --ndb-tls-search-path` was not correctly formatted. (Bug #36132430)

- **NDB Client Programs:** On Windows hosts, `ndb_sign_keys` could not locate the `ssh` program. (Bug #36053948)

- **NDB Client Programs:** `ndb_sign_keys` did not handle the `--CA-tool` option correctly on Windows. (Bug #36053908)

- **NDB Client Programs:** The use of a strict 80-character limit for `clang-format` on the file `CommandInterpreter.cpp` broke the formatting of the interactive help text in the `NDB` management client. (Bug #36034395)

- **NDB Client Programs:** Trying to start `ndb_mgmd` with `--bind-address=localhost` failed with the error `Illegal bind address`, which was returned from the MGM API when attempting to parse the bind adress to split it into host and port parts. `localhost` is now accepted as a valid address in such cases. (Bug #36005903)

- The included `libexpat` library was updated to version 2.5.0. (Bug #36324146)

- An implicit rollback generated when refusing to discover a table in an ongoing transaction caused the entire transaction to roll back. This could happen when a table definition changed while a transaction was active. We also checked at such times to see whether the table already existed in the data dictionary, which also meant that a subsequent read from same table within the same transaction would (wrongly) allow discovery.

  Now in such cases, we skip checking whether or not a given table already exists in the data dictionary; instead, we now always refuse discovery of a table that is altered while a transaction is ongoing and return an error to the user. (Bug #36191370)

- When a backup was restored using `ndb_restore` with `--disable-indexes` and `--restore-privilege-tables`, the ordered index of the primary key was lost on the `mysql.ndb_sql_metadata` table, and could not be rebuilt even with `--rebuild-indexes`. (Bug #36157626)

- `NDB` maintains both a local and a global pool of free send buffers. When send buffers cannot be allocated from the local pool NDB allocates one from the global pool; likewise, buffers are freed and returned to the global pool when the local pool has too many free buffers. Both of these allocations require a mutex to be locked.

In order to reduce contention on this global mutex, we attempt to over-allocate buffers from the global pool when needed, keeping the excess buffers in the local pool, when releasing excess buffers to the global pool this was done only to the limit determined by `max_free`. After having released to the global pool, such that the `max_free` limit was met, it was likely that additional buffers would soon be released, once again exceeding `max_free`. This caused extra contention on the global pool mutex.

To address this issue, we now reduce the free buffers to 2/3 of the `max_free` limit in such cases. (Bug #36108639)

- `SSL_pending()` data from an SSL-enabled `NdbSocket` was not adequately checked for. (Bug #36076879)
- In certain cases, `ndb_mgmd` hung when attempting to sending a stop signal to `ndbmtd`. (Bug #36066725)
- Starting a replica to apply changes when NDB was not yet ready or had no yet started led to an unhelpful error message (`Fatal error: Failed to run 'applier_start' hook`). This happened when the replica started and the applier start hook waited for the number of seconds specified by `--ndb-wait-setup` for NDB to become ready; if it was not ready by then, the start hook reported the failure. Now in such cases, we let processing continue, instead, and allow the error to be returned from NDB, which better indicates its true source. (Bug #36054134)
- A `mysqld` process took much longer than expected to shut down when all data nodes were unreachable. (Bug #36052113)
- Negated the need for handling in the `NDB` binary log injector thread for a failure to instantiate an injector transaction by removing a potential point of failure in that operation. (Bug #36048889)
- It was possible in certain cases for the `TRPMAN` block to operate on transporters outside its own receive thread. (Bug #36028782)
- Removed a possible race condition between `start_clients_thread()` and `update_connections()`, due to both of these seeing the same transporter in the `DISCONNECTING` state. Now we make sure that disconnection is in fact completed before we set indicating that that the transporter has disconnected, so that `update_connections()` cannot close the `NdbSocket` before it has been completely shut down. (Bug #36009860)
- When a transporter was overloaded, the send thread did not yield to the CPU as expected, instead retrying the transporter repeatedly until reaching the hard-coded 200 microsecond timeout. (Bug #36004838)
- A MySQL server disconnected from schema distribution was unable to set up event operations because the table columns could not be found in the event. This could be made to happen by using `ndb_drop_table` or another means to drop a table directly from `NDB` that had been created using the MySQL server.

  We fix this by making sure in such cases that we properly invalidate the `NDB` table definition from the dictionary cache. (Bug #35948153)
- The `ndb_sign_keys` utility's `--remote-openssl` option did not function as expected. (Bug #35853405)
- A replica could not apply a row change while handling a `Table definition changed` error. Now any such error is handled as a temporary error which can be retried multiple times. (Bug #35826145)
- Repeated incomplete incomplete attempts to perform a system restart in some cases left the cluster in a state from which it could not recover without restoring it from backup. (Bug #35801548)
- The event buffer used by the NDB API maintains an internal pool of free memory to reduce the interactions with the runtime and operating system, while allowing memory that is no longer needed

to be returned for other uses. This free memory is subtracted from the total allocated memory to determine the memory is use which is reported and used for enforcing buffer limits and other purposes; this was represented using a 32-bit value, so that if it exceeded 4 GB, the value wrapped, and the amount of free memory appeared to be reduced. This had potentially adverse effects on event buffer memory release to the runtime and OS, free memory reporting, and memory limit handling.

This is fixed by using a 64-bit value to represent the amount of pooled free memory. (Bug #35483764)

References: See also: Bug #35655162, Bug #35663761.

- `START REPLICA`, `STOP REPLICA`, and `RESET REPLICA` statements are now written to `mysqld.log`. (Bug #35207235)
- `NDB` transporter handling in `mt.cpp` differentiated between neighbor transporters carrying signals between nodes in the same node group, and all other transporters. This sometimes led to issues with multiple transporters when a transporter connected nodes that were neighbors with nodes that were not. (Bug #33800633)
- Removed unnecessary warnings generated by transient disconnections of data nodes during restore operations. (Bug #33144487)
- During setup of utility tables, the schema event handler sometimes hung waiting for the global schema lock (GSL) to become available. This could happen when the physical tables had been dropped from the cluster, or when the connection was lost for some other reason. Now we use a try lock when attempting to acquire the GSL in such cases, thus causing another setup check attempt to be made at a later time if the global schema lock is not available. (Bug #32550019, Bug #35949017)
- API nodes did not record any information in the log relating to disconnects due to missed heartbeats from the data nodes. (Bug #29623286)

Release Series Changelogs: MySQL NDB Cluster 8.4

This section contains unified changelog information for the NDB Cluster 8.4 release series.

For changelogs covering individual MySQL NDB Cluster 8.4 releases, see NDB Cluster Release Notes.

For general information about features added in MySQL NDB Cluster 8.4, see What is New in NDB Cluster 8.4.

For an overview of features added in MySQL 8.4 that are not specific to NDB Cluster, see What Is New in MySQL 8.4 since MySQL 8.0. For a complete list of all bug fixes and feature changes made in MySQL 8.4 that are not specific to NDB Cluster, see the MySQL 8.4 Release Notes.

Changes in MySQL NDB Cluster 8.4.8 (2026-01-23)

Note

These release notes were created with the assistance of MySQL HeatWave GenAI.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Reporting of File Close errors has been improved. (Bug #38726643)

- The option `--skip-fk-checks` was added to `ndb_restore`. This option modifies the behavior of `--rebuild-indexes` so that, when foreign keys are re-enabled, the existing data in the table is not checked for consistency. (Bug #38593666)
- The logs generated for backup and restore operations are improved in this release. (Bug #38592288)

Bugs Fixed

- NDB data nodes failed to start if either of the `NDBCNTR` or `DBDIH` system files were empty. (Bug #38424959)
- A data node failure occurred due to a reference validity check which failed. (Bug #38422448)
- BackupRecords in a participant node were not released from the pool when a backup was aborted, causing subsequent backup operations to fail. (Bug #38151265)
- Certain Transporter error messages did not include information on the source node. (Bug #37922543)
- The `ndb_restore` log message for total log bytes restored was incorrect (Bug #37697971)
- In the NDB management server, the TLS INFO statistics `upgraded` and `tls` were incremented if TLS authentication had failed. They should only be incremented on success. (Bug #36414579)

Changes in MySQL NDB Cluster 8.4.7 (2025-10-24)

Note

These release notes were created with the assistance of MySQL HeatWave GenAI.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Data node logging is improved with the following:
  - Local logs for data node join and leave events.
  - API disconnection handling logs.

  (Bug #38414245)
- MySQL NDB Cluster now generates log entries for redo logging issues, including `redo log buffer exhaustion`, `redo log space exhaustion`, and `file change problems`, which can cause transactions to be aborted, queued, or delayed. A secondary overload control mechanism monitors the rate at which the redo log part's Redo buffer is flushed to disk and estimates the time required to complete writing all data in the part's Redo buffer. (Bug #37903091)
- `LogLevelBackup` and `LogLevelSchema` Data node options were added in this release. (Bug #28004136)

Bugs Fixed

- If a data node restarted during a backup, the remaining data nodes could hang and the backup process stopped. The backup was marked as failed, and other nodes did not terminate their backup process, preventing further backups from being initiated. (Bug #38316252)
- It was not possible to restore backups with `BackupID` values greater than 2147483647, with the `ndb_restore` tool. Errors were returned similar to the following:

```
ndb_restore: [Warning] option 'backupid': signed value
3000000000 adjusted to 2147483647. Failed to find backup
2147483647.
```

(Bug #38260769)

- When running CREATE or INSERT statements on a NDB Binlog server with sql_log_bin = 0, replication would fail due to a table not existing. (Bug #37953883)
- Setting log level options for data nodes in MySQL NDB Cluster, such as `LogLevelCheckpoint`, `LogLevelStatistic`, `LogLevelInfo`, `LogLevelCongestion`, `LogLevelError`, and `LogLevelConnection` to a higher value does not provide additional log information beyond their default values. Errors were returned similar to the following:

```
Ndb kernel thread 0 is stuck in: Job Handling elapsed=100,
Watchdog: Warning overslept
```

(Bug #17847063)

Changes in MySQL NDB Cluster 8.4.6 (2025-07-23)

Note

These release notes were created with the assistance of MySQL HeatWave GenAI.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Timestamps in `NDB` node logs can now be printed with microsecond resolution. Data nodes can enable this feature using the data node `--ndb-log-timestamps=UTC` option; management nodes can also do so using the `ndb_mgmd` option `--ndb-log-timestamps=UTC`. For backwards compatible behavior, you can set this option explicitly to `LEGACY`, which uses the system time zone and resolution in seconds, as in previous releases. In NDB 8.4, this is the default.

  For SQL nodes, use the `--log-timestamps` option; be aware that this `mysqld` option does not support `LEGACY` as a value.

  See NDB Cluster Log Messages, for more information. (Bug #37924338)

Bugs Fixed

- **NDB Client Programs:** `ndb_restore`, when applying the log, allowed an infinite number of retries due to temporary errors, rather than limiting these to 11 as expected. (Bug #37883579)

  References: This issue is a regression of: Bug #31546136.

- **NDB Client Programs:** When restoring from backup with `DefaultOperationRedoProblemAction=ABORT`, an error in data node handling of a redo log part overload condition resulted in an incorrect error code (626 in this case) being sent back to `ndb_restore`, which caused `ndb_restore` to exit prematurely since it did not expect such an error. (Bug #37687485)

  References: This issue is a regression of: Bug #13219, Bug #13930, Bug #13980.

- Improved password handling for file system encryption.

Our thanks to Axel Svensson for the contribution. (Bug #37909595)

- `CREATE TABLESPACE`, when the specified logfile group did not exist, was rejected with error `723 No such table existed`, which did not correctly identify the source of the problem with the SQL statement. Now in such cases, the server issues Error `789 Logfile group not found`. (Bug #37802388)

- Warning 1296 `The temporary named table ... already exists` showed the table and schema names in the wrong order.

  Our thanks to Axel Svensson for the contribution. (Bug #117918, Bug #37807409)

Changes in MySQL NDB Cluster 8.4.5 (2025-04-16)

- Compilation Notes
- Functionality Added or Changed
- Bugs Fixed

Compilation Notes

- The ability to build the source without NDB using the internal script `storage/ndb/compile-cluster` was adversely affected by work done in NDB 8.0.31 making the `ndbcluster` plugin part of its default build. (Bug #117215, Bug #37484376)

Functionality Added or Changed

- Added the `Ndb_schema_participant_count` status variable. This variable provides the count of MySQL servers which are currently participating in NDB Cluster schema change distribution. (Bug #37529202)

Bugs Fixed

- **ndbinfo Information Database:** Certain queries against `ndbinfo` tables were not handled correctly. (Bug #37372650)

- **NDB Client Programs:** With a data node in an unstarted state, such as immediately after executing *`node_id`* `RESTART -n` in the `ndb_mgm` client, issuing `ALL REPORT BACKUPSTATUS` in the client subsequently led to an unplanned shutdown of the cluster. (Bug #37505513)

- **MySQL NDB ClusterJ:** The ClusterJ log file only reported the configured, requested node ID for a cluster connection (which was often zero). With this fix, after a connection has been established, ClusterJ reports the actual assigned node ID in the log. (Bug #37556172)

- **MySQL NDB ClusterJ:** A potential circular reference from `NdbRecordSmartValueHandlerImpl` that can cause delays in garbage collection has been removed. (Bug #37361267)

- **MySQL NDB ClusterJ:** Setting the connection property `com.mysql.clusterj.byte.buffer.pool.sizes` to "`512, 51200`" caused ClusterJ application to fail with a fatal exception thrown by `java.nio.ByteBuffer`. (Bug #37188154)

- **MySQL NDB ClusterJ:** When using a debug build of ClusterJ to run any tests in the testsuite, it exited with the error "1 thread(s) did not exit." (Bug #36383937)

- **MySQL NDB ClusterJ:** Running a ClusterJ application with Java 10 resulted in `java.lang.ClassNotFoundException`, because the class `java.internal.ref.Cleaner` is not available in Java 10. With this fix, the `java.lang.ref.Cleaner` class is used instead for resource cleanup. (Bug #29931569)

- The bundled `libxml2` library has been upgraded to version 2.9.13. (Bug #37806165)

- API node failure is detected by one or more data nodes; data nodes detecting API node failure inform all other data nodes of the failure, eventually triggering API node failure handling on each data node.

  Each data node handles API node failure independently; once all internal blocks have completed cleanup, the API node failure is considered handled, and, after a timed delay, the `QMGR` block allows the failed API node's node ID to be used for new connections.

  `QMGR` monitors API node failure handling, periodically generating warning logs for API node failure handling that has not completed (approximately every 30 seconds). These logs indicate which blocks have yet to complete failure handling.

  This enhancement improves logging in handling stalls particularly with regard to the `DBTC` block, which must roll back or commit and complete the API node's transactions, and release the associated `COMMIT` and `ACK` markers. In addition, the time to wait for API node failure handling is now configurable as the `ApiFailureHandlingTimeout` data node configuration parameter; after this number of seconds, handling is escalated to a data node restart. (Bug #37524092)

  References: See also: Bug #37469364.

- When a data node hangs during shutdown reasons for this may include: I/O problems on the node, in which case the thread shutting down hangs while operating on error and trace files; or an error in the shutdown logic, where the thread shutting down raises a Unix signal, and causes a deadlock. When such issues occur, users might observe watchdog warnings in the logs, referring to the last signal processed; this could be misleading in cases where there was actually a (different) preceding cause which had triggered the shutdown.

  To help pinpoint the origin of such problems if they occur, we have made the following improvements:

  - Added a new watchdog state `shutting down`. This is set early enough in the error handling process that it causes all watchdog logging of shutdown stalls to attribute the delay to a shutdown delay (correctly) rather than problem in execution.
  - We have also modified the watchdog mechanism to be aware of shutdown states, and use a more direct path—which is less likely to stall—to force the data node process to stop when needed.

  (Bug #37518267)

- When restoring `NDB` tables from backup, it is now possible for `mysqld` to open such tables even if their indexes are not yet available. (Bug #37516858)
- Signal dump code run when handling an unplanned node shutdown sometimes exited unexpectedly when speculatively reading section IDs which might not be present. (Bug #37512526)
- The LQH_TRANSCONF signal printer did not validate its input length correctly, which could lead the node process to exit. (Bug #37512477)
- When restoring stored grants (using `ndb_restore --include-stored-grants`) from an `NDB` backup following an initial restart of the data nodes, the `ndb_sql_metadata` table was neither created nor restored. (Bug #37492169)
- Nothing was written to the cluster log to indicate when `PURGE BINARY LOGS` had finished waiting for the purge to complete. (Bug #37489870)
- `WITH_NDB_TLS_SEARCH_PATH` was not set when compiling NDB Cluster using `WITHOUT_SERVER`. (Bug #37398657)
- Now, when `ndb_metadata_check` is enabled, we synchronize both schema and tables in the same interval. (Bug #37382551)
- This fix addresses the following two issues:

1. When a resend could not be started due to a gap created by an out of buffer error in the event stream, forwarding event data for the bucket was not initiated. We fix this by ensure that the takeover process has been initiated before exiting the resend code.

2. An out of buffer error which occurred during an ongoing resend was not handled. In this case, we now interrupt the resend when such an error is raised.

(Bug #37349305)

- On certain rare occasions, when concurrent calls were made to `release()` and `get()`, instances of `ndb_schema_object` were doubly freed. (Bug #35793818)

- If an out-of-buffer-release (OOBR) process took an excessive amount of time, the reset was performed prematurely, before all buffers were released, thus interfering with concurrent seizing of new pages, beginning new `out_of_buffer` handling, or starting a resend.

  We solve this issue by ensuring that resumption of event buffering takes place only after the OOBR process has completed for all buckets. (Bug #20648778)

Changes in MySQL NDB Cluster 8.4.4 (2025-01-22)

- Compilation Notes
- Bugs Fixed

Compilation Notes

- **macOS:** A `uint64_t` value used with `%zu` caused a `[-Wformat]` compiler warning on MacOS. (Bug #37174692)

- Removed a warning in `storage/ndb/src/common/util/cstrbuf.cpp`. (Bug #37049014)

Bugs Fixed

- **Microsoft Windows:** Successive iterations of the sequence `ndb_sign_keys --create-key` followed by `ndb_sign_keys --promote` were unsuccessful on Windows. (Bug #36951132)

- **NDB Disk Data:** `mysqld` did not use a disk scan for `NDB` tables with 256 disk columns or more. (Bug #37201922)

- **NDB Cluster APIs:** The `Ndb_cluster_connection` destructor calls `g_eventLogger::stopAsync()` in order to release the buffers used by the asynchronous logging mechanism as well as to stop the threads responsible for this logging. When the `g_eventLogger` object was deleted before the `Ndb_cluster_connection` destructor was called, the application terminated after trying to use a method on a null object. This could happen in either of two ways:

  - An API program deleted the logger object before deleting the `Ndb_cluster_connection`.

  - `ndb_end()` was called before the `Ndb_cluster_connection` was deleted.

  We solve this issue by skipping the call to `stopAsync()` in the `Ndb_cluster_connection` destructor when `g_eventLogger` is `NULL`. This fix also adds a warning to inform API users that deleting `g_eventLogger` before calling the `Ndb_cluster_connection` destructor is incorrect usage.

  For more information, see API Initialization and Cleanup. (Bug #37300558)

- **NDB Cluster APIs:** Removed known causes of API node versus data node state misalignments, and improved the handling of state misalignments when detected. In one such case, separate handling of scan errors in the `NDB` kernel and those originating in API programs led to cleanup not being

performed after some scans. Handling of DBTC and API state alignment errors has been improved by this set of fixes, as well as scan protocol timeout handling in DBSPJ; now, when such misalignments in state are detected, the involved API nodes are disconnected rather than the data node detecting it being forced to shut down. (Bug #20430083, Bug #22782511, Bug #23528433, Bug #28505289, Bug #36273474, Bug #36395384, Bug #36838756, Bug #37022773, Bug #37022901, Bug #37023549)

References: See also: Bug #22782511, Bug #23528433, Bug #36273474, Bug #36395384, Bug #36838756.

- **ndbinfo Information Database:** At table create and drop time, access of ndbinfo tables such as operations_per_fragment and memory_per_fragment sometimes examined data which was not valid.

  To fix this, during scans of these ndbinfo tables, we ignore any fragments from tables in transient states at such times due to being created or dropped. (Bug #37140331)

- Work done previously to support opening NDB tables with missing indexes was intended to allow the features of the MySQL server to be used to solve problems in cases where indexes cannot be rebuilt due to unmet constraints. With missing indexes, some of the SQL handler functionality is unavailable —for example, the use of indexes to select rows for modification efficiently, or to identify duplicates when processing modifications, or to push joins relying on indexes. This could lead to the unplanned shutdown of an NDB Cluster SQL node.

  In such cases, the server now simply returns an error. (Bug #37299071)

- Recent refactoring of the transporter layer added the reporting of the presence of socket shutdown errors, but not their nature. This led to confusion in the common case where a socket shutdown is requested, but the socket is already closed by the peer. To avoid such confusion, this logging has been removed. (Bug #37243135)

  References: This issue is a regression of: Bug #35750771.

- It was not possible to create an NDB table with 256 or more BLOB columns when also specifying a reduced inline size, as in the following SQL statement:

```
CREATE TABLE t1 (
   pk INT PRIMARY KEY,
   b1 BLOB COMMENT 'NDB_COLUMN=BLOB_INLINE_SIZE=100',
   b2 BLOB COMMENT 'NDB_COLUMN=BLOB_INLINE_SIZE=100',
   ...,
   b256 BLOB COMMENT 'NDB_COLUMN=BLOB_INLINE_SIZE=100'
) ENGINE=NDBCLUSTER;
```

  (Bug #37201818)

- In some cases, the occurrence of node failures during shutdown led to the cluster becoming unrecoverable without manual intervention.

  We fix this by modifying global checkpoint ID (GCI) information propagation (CopyGCI mechanism) to reject propagation of any set of GCI information which does not describe the ability to recover the cluster automatically as part of a system restart. (Bug #37163647)

  References: See also: Bug #37162636.

- In some cases, node failures during an otherwise graceful shutdown could lead to a cluster becoming unrecoverable without manual intervention. This fix modifies the generic GCI info propagation mechanism (CopyGCI) to reject propagating any set of GCI information which does not describe the ability to recover a cluster automatically. (Bug #37162636)

- Improved variable names used in start_resend(), and enhanced related debug messages to users and developers with additional information. (Bug #37157987)

- In certain cases, a COPY_FRAGREQ signal did not honor a fragment scan lock. (Bug #37125935)

- In cases where NDB experienced an API protocol timeout when attempting to close a scan operation, it considered the DBTC ApiConnectRecord involved to be lost for further use, at least until the API disconnected and API failure handling within DBTC reclaimed the record.

  This has been improved by having the API send a TCRELEASEREQ signal to DBTC in such cases, performing API failure handling for a single ApiConnectRecord within DBTC. (Bug #37023661)

  References: See also: Bug #36273474, Bug #36395384, Bug #37022773, Bug #37022901, Bug #37023549.

- For tables using the NDB storage engine, the column comment option BLOB_INLINE_SIZE was silently ignored for TINYBLOB columns, and (silently) defaulted to the hard-coded 256 byte value regardless of the size provided; this was misleading to users.

  To fix this problem, we now specifically disallow BLOB_INLINE_SIZE on TINYBLOB columns altogether, and NDB now prints a warning saying that the column size is defaulting to 256 bytes. (Bug #36725332)

- Testing revealed that a fix for a previous issue which added a check of the ApiConnectRecord failure number against the system's current failure number did not initialize the ApiConnectRecord failure number in all cases. (Bug #36155195)

  References: This issue is a regression of: Bug #36028828.

- ndb_config did not always handle very long file paths correctly.

  Our thanks to Dirkjan Bussink for the contribution. (Bug #116748, Bug #37310680)

- Errors of unknown provenance were logged while assigning node IDs during cluster synchronization, leading to user doubt and concern. Logging of the data node QMGR block and the ndb_mgmd process relating to node ID allocation issues has therefore been improved, to supply more and better information about what is being reported in such cases. (Bug #116351, Bug #37189356)

- A multi-range scan sometimes lost its fragment lock for the second and subsequent ranges of the scan. (Bug #111932, Bug #35660890)

Changes in MySQL NDB Cluster 8.4.3 (2024-10-16)

- Deprecation and Removal Notes
- Functionality Added or Changed
- Bugs Fixed

Deprecation and Removal Notes

- **NDB Client Programs:** The ndb_size.pl utility is now deprecated and is no longer supported. You can expect it to be removed from a future version of the NDB Cluster distribution; for this reason, you should now modify any applications which depend on it accordingly. (WL #16456)

- Use of an Ndb.cfg file for setting the connection string for an NDB process was not well documented or supported. With this release, this file is now formally deprecated, and you should expect support for it to be removed in a future release of MySQL Cluster. (WL #15765)

Functionality Added or Changed

- The ndbcluster plugin subscribes to all changes that occur in NDB and writes them epoch by epoch to the binary log. Each epoch received from NDB consists of a large number of changes, all of which are written to the binary log transaction cache before flushing them to the binary log. Previously, it was possible to configure the cache size for all threads, which often led to improper resource allocation for a MySQL Server used for writing a binary log of changes for NDB.

To enable dimensioning and configuring the system properly, we introduce a new system variable `ndb_log_cache_size` which makes it possible to set the size of the transaction cache used by the `NDB` binary log injector, so that this size can be set separately for writing the binary log for NDB transactions and (using `binlog_cache_size`) for writing other transactions whose sizes are likely to be smaller. (Bug #36694848)

Bugs Fixed

- **NDB Cluster APIs:** Using `NdbRecord` and `OO_SETVALUE` from the NDB API to write the value of a `Varchar`, `Varbinary`, `Longvarchar`, or `Longvarbinary` column failed with error `829`. (Bug #36989337)
- **MySQL NDB ClusterJ:** References to ClusterJPA and OpenJPA have been removed from the comments in the packaging files, as JPA code was already removed from ClusterJ some time ago. (Bug #36725675)
- **MySQL NDB ClusterJ:** `ReconnectTest` in the ClusterJ test suite failed sometimes due to a race condition. The test has been rewritten with proper synchronization. (Bug #28550140)
- Removed node management code from `TRIX` that was not actually used. (Bug #37006547)
- Submitting concurrent shutdown commands for individual nodes using `ndb_mgm SHUTDOWN` *`node_id`* or the MGM API sometimes had one or both of the following adverse results:
  - Cluster failure when all nodes in the same node group were stopped
  - Inability to recover when all nodes in the same node group were stopped, and the cluster had more than one node group

  This was due to the fact that the (planned) shutdown of a single node assumed that only one such shutdown occurred at a time, but did not actually check this limitation.

  We fix this so that concurrent single-node shutdown requests are serialized across the cluster, and any which would cause a cluster outage are rejected. (Bug #36943756)

  References: See also: Bug #36839995.
- Shutdown of a data node late in a schema transaction updating index statistics caused the president node to shut down as well. (Bug #36886242)

  References: See also: Bug #36877952.
- It was possible for duplicate events to be sent to user applications when a data node was shut down. (Bug #36750146)
- `BLOB_INLINE_SIZE=0` set within a column comment was not honored, and the default for the blob type was used instead (such as 256 bytes for `BLOB`).

  See NDB_COLUMN Options, for more information. (Bug #36724336)
- Issues arose when an attempt was made to use a SHM transporter's wakeup socket before it was ready, due in part to error-handling when setting up the SHM transporter, which did not close the socket correctly prior to making another attempt at setup. (Bug #36568752, Bug #36623058)
- An error in a `my.cnf` file could cause the management node to shut down unexpectedly. (Bug #36508565)
- A race condition sometimes occurred between the watchdog thread and the signal execution thread trying to start node failure handling in parallel. (Bug #35728261)

Changes in MySQL NDB Cluster 8.4.1 (2024-07-02)

Important

This release is no longer available for download. It was removed due to a critical issue that could stop the server from restarting following the creation of a very large number of tables (8001 or more). Please upgrade to MySQL Cluster 8.4.2 instead.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change:** Now, when the removal of a data node file or directory fails with a file does not exist (**ENOENT**) error, this is treated as a successful removal.
- **ndbinfo Information Database:** Added a `type` column to the `transporter_details` table in the `ndbinfo` information database. This column shows the type of connection used by the transporter, which is either of `TCP` or `SHM`.
- **NDB Client Programs:** Added the `--CA-days` option to `ndb_sign_keys` to make it possible to specify a certificate's lifetime. (Bug #36549567)
- **NDB Client Programs:** When started, `ndbd` now produces a warning in the data node log like this one:

  ```
  2024-05-28 13:32:16 [ndbd] WARNING  -- Running ndbd with a single thread of
  signal execution.  For multi-threaded signal execution run the ndbmtd binary.
  ```

  (Bug #36326896)

Bugs Fixed

- **NDB Cluster APIs:** It was possible to employ the following NDB API methods without them being used as `const`, although this alternative usage had long been deprecated (and was not actually documented):

  - `Dictionary::listEvents()`
  - `Dictionary::listIndexes()`
  - `Dictionary::listObjects()`
  - `NdbOperation::getNdbErrorLine()`

  Now, each of these methods must always be invoked as `const`. (Bug #36165876)
- **NDB Client Programs:** `ndb_redo_log_reader` could not read data from encrypted files. (Bug #36313482)
- **NDB Client Programs:** `ndb_redo_log_reader` exited with `Record type = 0 not implemented` when reaching an unused page, all zero bytes, or a page which was only partially used (typically a page consisting of the page header only). (Bug #36313259)
- **NDB Client Programs:** `ndb_restore` did not restore a foreign key whose columns differed in order from those of the parent key.

  Our thanks to Axel Svensson for the contribution. (Bug #114147, Bug #36345882)

- The destructor for `NDB_SCHEMA_OBJECT` makes several assertions about the state of the schema object, but the state was protected by a mutex, and the destructor did not acquire this mutex before testing the state.

  We fix this by acquiring the mutex within the destructor. (Bug #36568964)

- `NDB` now writes a message to the MySQL server log before and after logging an incident in the binary log. (Bug #36548269)

- Removed a memory leak in `/util/NodeCertificate.cpp`. (Bug #36537931)

- Removed a memory leak from `src/ndbapi/NdbDictionaryImpl.cpp`. (Bug #36532102)

- The internal method `CertLifetime::set_set_cert_lifetime(X509 *cert)` should set the not-before and not-after times in the certificate to the same as those stored in the `CertLifetime` object, but instead it set the not-before time to the current time, and the not-after time to be of the same duration as the object. (Bug #36514834)

- Removed a possible use-after-free warning in `ConfigObject::copy_current()`. (Bug #36497108)

- When a thread acquires and releases the global schema lock required for schema changes and reads, the associated log message did not identify who performed the operation.

  To fix this issue, we now do the following:

  - Prepend the message in the log with the identification of the NDB Cluster component or user session responsible.
  - Provide information about the related Performance Schema thread so that it can be traced.

  (Bug #36446730)

  References: See also: Bug #36446604.

- Metadata changes were not logged with their associated thread IDs. (Bug #36446604)

  References: See also: Bug #36446730.

- When building `NDB` using `lld`, the build terminated prematurely with the error message `ld.lld: error: version script assignment of 'local' to symbol 'my_init' failed: symbol not defined` while attempting to link `libndbclient.so`. (Bug #36431274)

- TLS did not fail cleanly on systems which used OpenSSL 1.0, which is unsupported. Now in such cases, users get a clear error message advising that an upgrade to OpenSSL 1.1 or later is required to use TLS with NDB Cluster. (Bug #36426461)

- NDB Cluster's pushdown join functionality expects pushed conditions to filter exactly, so that no rows that do not match the condition must be returned, and all rows that do match the condition must returned. When the condition contained a BINARY value compared to a `BINARY` column this was not always true; if the value was shorter than the column size, it could compare as equal to a column value despite having different lengths, if the condition was pushed down to `NDB`.

  Now, when deciding whether a condition is pushable, we also make sure that the `BINARY` value length exactly matches the `BINARY` column's size. In addition, when binary string values were used in conditions with `BINARY` or `VARBINARY` columns, the actual length of a given string value was not used but rather an overestimate of its length. This is now changed; this should allow more conditions comparing short string values with `VARBINARY` columns to be pushed down than before this fix was made. (Bug #36390313, Bug #36513270)

  References: See also: Bug #36399759, Bug #36400256. This issue is a regression of: Bug #36364619.

- Setting `AutomaticThreadConfig` and `NumCPUs` when running single-threaded data nodes (`ndbd`) sometimes led to unrecoverable errors. Now `ndbd` ignores settings for these parameters, which are intended to apply only to multi-threaded data nodes (`ndbmtd`). (Bug #36388981)

- Improved the error message returned when trying to add a primary key to an `NDBCLUSTER` table using `ALGORITHM=INPLACE`. (Bug #36382071)

  References: See also: Bug #30766579.

- The handling of the `LQH` operation pool which occurs as part of TC takeover skipped the last element in either of the underlying physical pools (static or dynamic). If this element was in use, holding an operation record for a transaction belonging to a transaction coordinator on the failed node, it was not returned, resulting in an incomplete takeover which sometimes left operations behind. Such operations interfered with subsequent transactions and the copying process (`CopyFrag`) used by the failed node to recover.

  To fix this problem, we avoid skipping the final record while iterating through the `LQH` operation records during TC takeover. (Bug #36363119)

- When distribution awareness was not in use, the cluster tended to choose the same data node as the transaction coordinator repeatedly. (Bug #35840020, Bug #36554026)

- In certain cases, management nodes were unable to allocate node IDs to restarted data and SQL nodes. (Bug #35658072)

- Setting `ODirect` in the cluster's configuration caused excess logging when verifying that `ODirect` was actually settable for all paths. (Bug #34754817)

- In some cases, when trying to perform an online add index operation on an `NDB` table with no explicit primary key (see Limitations of NDB online operations), the resulting error message did not make the nature of the problem clear. (Bug #30766579)

  References: See also: Bug #36382071.

Changes in MySQL NDB Cluster 8.4.0 (2024-04-30)

- Deprecation and Removal Notes
- Functionality Added or Changed
- Bugs Fixed

Deprecation and Removal Notes

- **Packaging; Linux:** Removed the deprecated tool `/usr/bin/pathfix.py` from packages for Fedora 39. (Bug #35997178)

Functionality Added or Changed

- **ndbinfo Information Database:** Added the `transporter_details` table to the `ndbinfo` information database. This table is similar to the `transporters` table, but provides information about individual transporters rather than in the aggregate.

  For more information, see The ndbinfo transporter_details Table. (Bug #113163, Bug #36031560)

- **NDB Client Programs:** Added the `--verbose` option to the `ndb_waiter` test program to control the verbosity level of the output. (Bug #34547034)

- Improved logging related to purging of the binary log, including start and completions times, and whether it is the injector which has initiated the purge. (Bug #36176983)

Bugs Fixed

- **NDB Cluster APIs:** TLS connection errors were printed even though TLS was not specified for connections.

  To fix this issue, following an ignored TLS error, we explicitly reset the error condition in the management handle to **NO_ERROR**. (Bug #36354973)

- **NDB Cluster APIs:** The `NdbEventOperation` methods `hasError()` and `clearError()`, long deprecated, are effectively disabled: `hasError()` now returns a constant 0, and `clearError()` does nothing. To determine an event type, use `getEventType2()` instead.

- **NDB Client Programs:** In some cases, it was not possible to load cerificates generated using `ndb_sign_keys`. (Bug #36430004)

- **NDB Client Programs:** The following command-line options did not function correctly for the `ndb_redo_log_reader` utility program:

  - `--mbyte`
  - `--page`
  - `--pageindex`

  (Bug #36313427)

- **NDB Client Programs:** A certificate lifetime generated by `ndb_sign_keys` should consist of a fixed number of days, plus a random amount of extra time provided by the OpenSSL function `RAND_bytes()`, casting the result to a signed integer value. Because this value could sometimes be negative, this led to extra time being subtracted rather than added.

  We eliminate this problem by using an unsigned integer type to hold the value obtained from `RAND_bytes()`. (Bug #36270629)

- **NDB Client Programs:** Invoking ndb_mgmd with the --bind-address option could in some cases cause the program to terminate unexpectedly. (Bug #36263410)

- **NDB Client Programs:** Some `NDB` utilities such `ndb_show_tables` leaked memory from API connections when TLS was required by the data nodes, and with valid certificates. (Bug #36170703)

- **NDB Client Programs:** Work begun in NDB 8.0.18 and 8.0.20 to remove the unnecessary text `NDBT_ProgramExit ...` from the output of NDB programs is completed in this release. This message should no longer appear in the release binaries of any such programs. (Bug #36169823)

  References: See also: Bug #27096741.

- **NDB Client Programs:** The output from `ndb_waiter --ndb-tls-search-path` was not correctly formatted. (Bug #36132430)

- **NDB Client Programs:** On Windows hosts, `ndb_sign_keys` could not locate the `ssh` program. (Bug #36053948)

- **NDB Client Programs:** `ndb_sign_keys` did not handle the `--CA-tool` option correctly on Windows. (Bug #36053908)

- **NDB Client Programs:** The use of a strict 80-character limit for `clang-format` on the file `CommandInterpreter.cpp` broke the formatting of the interactive help text in the `NDB` management client. (Bug #36034395)

- **NDB Client Programs:** Trying to start `ndb_mgmd` with `--bind-address=localhost` failed with the error `Illegal bind address`, which was returned from the MGM API when attempting to parse the bind adress to split it into host and port parts. `localhost` is now accepted as a valid address in such cases. (Bug #36005903)

- An implicit rollback generated when refusing to discover a table in an ongoing transaction caused the entire transaction to roll back. This could happen when a table definition changed while a transaction was active. We also checked at such times to see whether the table already existed in the data dictionary, which also meant that a subsequent read from same table within the same transaction would (wrongly) allow discovery.

  Now in such cases, we skip checking whether or not a given table already exists in the data dictionary; instead, we now always refuse discovery of a table that is altered while a transaction is ongoing and return an error to the user. (Bug #36191370)

- When a backup was restored using `ndb_restore` with `--disable-indexes` and `--restore-privilege-tables`, the ordered index of the primary key was lost on the `mysql.ndb_sql_metadata` table, and could not be rebuilt even with `--rebuild-indexes`. (Bug #36157626)

- `NDB` maintains both a local and a global pool of free send buffers. When send buffers cannot be allocated from the local pool NDB allocates one from the global pool; likewise, buffers are freed and returned to the global pool when the local pool has too many free buffers. Both of these allocations require a mutex to be locked.

  In order to reduce contention on this global mutex, we attempt to over-allocate buffers from the global pool when needed, keeping the excess buffers in the local pool, when releasing excess buffers to the global pool this was done only to the limit determined by `max_free`. After having released to the global pool, such that the `max_free` limit was met, it was likely that additional buffers would soon be released, once again exceeding `max_free`. This caused extra contention on the global pool mutex.

  To address this issue, we now reduce the free buffers to 2/3 of the `max_free` limit in such cases. (Bug #36108639)

- `SSL_pending()` data from an SSL-enabled `NdbSocket` was not adequately checked for. (Bug #36076879)

- In certain cases, `ndb_mgmd` hung when attempting to sending a stop signal to `ndbmtd`. (Bug #36066725)

- Starting a replica to apply changes when NDB was not yet ready or had no yet started led to an unhelpful error message (`Fatal error: Failed to run 'applier_start' hook`). This happened when the replica started and the applier start hook waited for the number of seconds specified by `--ndb-wait-setup` for NDB to become ready; if it was not ready by then, the start hook reported the failure. Now in such cases, we let processing continue, instead, and allow the error to be returned from NDB, which better indicates its true source. (Bug #36054134)

- A `mysqld` process took much longer than expected to shut down when all data nodes were unreachable. (Bug #36052113)

- Negated the need for handling in the `NDB` binary log injector thread for a failure to instantiate an injector transaction by removing a potential point of failure in that operation. (Bug #36048889)

- It was possible in certain cases for the `TRPMAN` block to operate on transporters outside its own receive thread. (Bug #36028782)

- Removed a possible race condition between `start_clients_thread()` and `update_connections()`, due to both of these seeing the same transporter in the `DISCONNECTING` state. Now we make sure that disconnection is in fact completed before we set indicating that that the transporter has disconnected, so that `update_connections()` cannot close the `NdbSocket` before it has been completely shut down. (Bug #36009860)

- When a transporter was overloaded, the send thread did not yield to the CPU as expected, instead retrying the transporter repeatedly until reaching the hard-coded 200 microsecond timeout. (Bug #36004838)

- A MySQL server disconnected from schema distribution was unable to set up event operations because the table columns could not be found in the event. This could be made to happen by using `ndb_drop_table` or another means to drop a table directly from `NDB` that had been created using the MySQL server.

  We fix this by making sure in such cases that we properly invalidate the `NDB` table definition from the dictionary cache. (Bug #35948153)

- The `ndb_sign_keys` utility's `--remote-openssl` option did not function as expected. (Bug #35853405)

- A replica could not apply a row change while handling a `Table definition changed` error. Now any such error is handled as a temporary error which can be retried multiple times. (Bug #35826145)

- Repeated incomplete incomplete attempts to perform a system restart in some cases left the cluster in a state from which it could not recover without restoring it from backup. (Bug #35801548)

- The event buffer used by the NDB API maintains an internal pool of free memory to reduce the interactions with the runtime and operating system, while allowing memory that is no longer needed to be returned for other uses. This free memory is subtracted from the total allocated memory to determine the memory is use which is reported and used for enforcing buffer limits and other purposes; this was represented using a 32-bit value, so that if it exceeded 4 GB, the value wrapped, and the amount of free memory appeared to be reduced. This had potentially adverse effects on event buffer memory release to the runtime and OS, free memory reporting, and memory limit handling.

  This is fixed by using a 64-bit value to represent the amount of pooled free memory. (Bug #35483764)

  References: See also: Bug #35655162, Bug #35663761.

- `START REPLICA`, `STOP REPLICA`, and `RESET REPLICA` statements are now written to `mysqld.log`. (Bug #35207235)

- `NDB` transporter handling in `mt.cpp` differentiated between neighbor transporters carrying signals between nodes in the same node group, and all other transporters. This sometimes led to issues with multiple transporters when a transporter connected nodes that were neighbors with nodes that were not. (Bug #33800633)

- Removed unnecessary warnings generated by transient disconnections of data nodes during restore operations. (Bug #33144487)

- During setup of utility tables, the schema event handler sometimes hung waiting for the global schema lock (GSL) to become available. This could happen when the physical tables had been dropped from the cluster, or when the connection was lost for some other reason. Now we use a try lock when attempting to acquire the GSL in such cases, thus causing another setup check attempt to be made at a later time if the global schema lock is not available. (Bug #32550019, Bug #35949017)

- API nodes did not record any information in the log relating to disconnects due to missed heartbeats from the data nodes. (Bug #29623286)

Changes in MySQL NDB Cluster 8.3.0 (2024-01-16)

- Compilation Notes
- Functionality Added or Changed
- Bugs Fixed

Compilation Notes

- **NDB Cluster APIs:** In MySQL 8.0 and later, it was necessary to build MGM API applications using a C++ compiler. In addition, the compiler requirements for both NDB API and MGM API applications were not consistent between NDB Cluster releases. This fix addresses both issues as follows:

  - MGM API applications now require a C compiler that supports C99 or later.
  - NDB API applications now require a compiler that supports C++11 or later.

  Pre-release testing has also been improved to ensure that future versions of the APIs continue to meet these requirements.

  For more detailed information about language support and compiler requirements for building NDB Cluster API applications, including those for previous versions of NDB, see General Requirements. (WL #15908)

- NDB Cluster did not compile correctly on Ubuntu 23.10. (Bug #35847193)

- It is now possible to build NDB Cluster for the s390x platform.

  Our thanks to Namrata Bhave for the contribution. (Bug #110807, Bug #35330936)

Functionality Added or Changed

- This release implements support for network communications between NDB nodes secured by Transport Layer Security (TLS) and Internet Public Key Infrastructure (PKI) to authenticate and encrypt connections, and between the NDB management server and its clients. TLS is applied both to the NDB Transporter Protocol, and to the NDB Management Protocol. In both cases, this is done using TLS mutual authentication.

  (Connections that use the MySQL client protocol employ MySQL user authentication which can use TLS; see Using Encrypted Connections, for more information.)

  A new tool `ndb_sign_keys` can be used to create and manage CA, certificate files, and keys. You can generate a set of keys and certificates for all nodes in a cluster using `ndb_sign_keys --create-key`.

  Private keys are created in place, so that copying of files containing private keys is minimized. Both private keys and certificates are labeled as either active or pending; `ndb_sign_keys` also provides help with rotating keys to allow for pending keys to replace active keys before the active keys expire.

  You can test node TLS connections with `ndb_mgm --test-tls`, or from within the `ndb_mgm` client using the `TLS INFO` command. You can also obtain information about certificates used by cluster nodes by checking the `ndbinfo certificates` table.

  You can enforce a requirement for TLS on the cluster, by setting the appropriate client options and node configuration parameters. See Using TLS Connections, for details.

  Use of TLS connections is also now supported in NDB Cluster API applications. For information about MGM API support, see TLS Functions. The NDB API now provides `configure_tls()` `get_tls_certificate_path()` methods of `Ndb_cluster_connection` for setting up TLS connections by clients.

  For more information, see TLS Link Encryption for NDB Cluster, and ndb_sign_keys — Create, Sign, and Manage TLS Keys and Certificates for NDB Cluster. (WL #15135, WL #15154, WL #15166, WL #15521)

Bugs Fixed

- **NDB Replication:** An internal thread memory usage self-check was too strict, invoking unnecessary file rotation and possibly increased memory usage. (Bug #35657932)

- **NDB Replication:** `CREATE USER` on a source cluster caused SQL nodes attached to the replica clusters to exit. (Bug #34551954)

  References: See also: Bug #112775, Bug #33172887, Bug #33542052, Bug #35928350.

- **NDB Replication:** Replicating a `GRANT NDB_STORED_USER` statement with replication filters enabled caused the SQL node to exit. This occured since the replication filter caused all non-updating queries to return an error, with the assumption that only changes needed to be replicated.

  Our thanks to Mikael Ronström for the contribution. (Bug #112775, Bug #35928350)

  References: See also: Bug #34551954, Bug #33172887, Bug #33542052.

- **NDB Replication:** On an NDB Replication setup where an SQL node in a replica cluster had `read_only=ON`, a `DROP DATABASE` statement on the source cluster caused the SQL thread on the replica server to hang with `Waiting for schema metadata lock`.

- **NDB Cluster APIs:** An event buffer overflow in the NDB API could cause a timeout while waiting for `DROP TABLE`. (Bug #35655162)

  References: See also: Bug #35662083.

- **ndbinfo Information Database:** An assumption made in the implementation of `ndbinfo` is that the data nodes always use the same table ID for a given table at any point in time. This requires that a given table ID is not moved between different tables in different versions of NDB Cluster, as this would expose an inconsistency during a rolling upgrade. This constraint is fairly easily maintained when `ndbinfo` tables are added only in the latest release, and never backported to a previous release series, but could be problematic in the case of a backport.

  Now we ensure that, if a given `ndbinfo` table added in a newer release series is later backported to an older one, the table uses the same ID as in the newer release. (Bug #28533342)

- When a node failure is detected, transaction coordinator (TC) instances check their own transactions to determine whether they need handling to ensure completion, implemented by checking whether each transaction involves the failed node, and if so, marking it for immediate timeout handling. This causes the transaction to be either rolled forward (commit) or back (abort), depending on whether it had started committing, using the serial commit protocol. When the TC was in the process of getting permission to commit (`CS_PREPARE_TO_COMMIT`), sending commit requests (`CS_COMMITTING`), or sending completion requests (`CS_COMPLETING`), timeout handling waited until the transaction was in a stable state before commencing the serial commit protocol.

  Prior to the fix for Bug#22602898, all timeouts during `CS_COMPLETING` or `CS_COMMITTING` resulted in switching to the serial commit-complete protocol, so skipping the handling in any of the three states cited previously did not stop the prompt handling of the node failure. It was found later that this fix removed the blanket use of the serial commit-complete protocol for commit-complete timeouts, so that when handling for these states was skipped, no node failure handling action was taken, with the result that such transactions hung in a commit or complete phase, blocking checkpoints.

  The fix for Bug#22602898 removed this stable state handling to avoid it accidentally triggering, but this change also stopped it from triggering when needed in this case where node failure handling found a transaction in a transient state. We solve this problem by modifying `CS_COMMIT_SENT` and `CS_COMPLETE_SENT` stable state handling to perform node failure processing if a timeout has occurred for a transaction with a failure number different from the current latest failure number,

ensuring that all transactions involving the failed node are in fact eventually handled. (Bug #36028828)

References: See also: Bug #22602898.

- The `QMGR` block's `GSN_ISOLATE_ORD` signal handling was modified by the fix for a previous issue to handle the larger node bitmap size necessary for supporting up to 144 data nodes. It was observed afterwards that it was possible that the original sender was already shut down when `ISOLATE_ORD` was processed, in which case its node version might have been reset to zero, causing the inline bitmap path to be taken, resulting in incorrect processing.

  The signal handler now checks to decide whether the incoming signal uses a long section to represent nodes to isolate, and to act accordingly. (Bug #36002814)

  References: See also: Bug #30529132.

- Messages like `Metadata: Failed to submit table 'mysql.ndb_apply_status' for synchronization` were submitted to the error log each minute, which filled up the log unnecessarily, since `mysql.ndb_apply_status` is a utility table managed by the binary logging thread, with no need to be checked for changes. (Bug #35925503)

- The `DBSPJ` function `releaseGlobal()` is responsible for releasing excess pages maintained in m_free_page_list; this function iterates over the list, releases the objects, and after 16 iterations takes a realtime break. In parallel with the realtime break, `DBSPJ` spawned a new invocation of `releaseGlobal()` by sending a `CONTINUEB` signal to itself with a delay, which could lead to an overflow of the Long-Time Queue since there is no control over the number of signals being sent.

  We fix this by not sending the extra delayed `CONTINUEB` signal when a realtime break is taken. (Bug #35919302)

- API node failure handling during a data node restart left its subscriptions behind. (Bug #35899768)

- Removed the file `storage/ndb/tools/restore/consumer_restorem.cpp`, which was unused. (Bug #35894084)

- Removed unnecessary output printed by `ndb_print_backup_file`. (Bug #35869988)

- Removed a possible accidental read or write on a reused file descriptor in the transporter code. (Bug #35860854)

- When a timed read function such as `read_socket()`, `readln_socket()`, `NdbSocket::read()`, or `NdbSocket::readln()` was called using an invalid socket it returned `0`, indicating a timeout, rather than the expected `-1`, indicating an unrecoverable failure. This was especially apparent when using the `poll()` function, which, as a result of this issue, did not treat an invalid socket appropriately, but rather simply never fired any event for that socket. (Bug #35860646)

- It was possible for the `readln_socket()` function in `storage/ndb/src/common/util/socket_io.cpp` to read one character too many from the buffer passed to it as an argument. (Bug #35857936)

- It was possible for `ssl_write()` to receive a smaller send buffer on retries than expected due to `consolidate()` calculating how many full buffers could fit into it. Now we pre-pack these buffers prior to consolidation. (Bug #35846435)

- During online table reorganization, rows that are moved to new fragments are tagged for later deletion in the copy phase. This tagging involves setting the `REORG_MOVED` bit in the tuple header; this affects the tuple header checksum which must therefore be recalculated after it is modified. In some cases this is calculated before `REORG_MOVED` is set, which can result in later access to the same tuple failing with a tuple header checksum mismatch. This issue was observed when executing `ALTER TABLE REORGANIZE PARTITION` concurrently with a table insert of blob values, and appears to have been a side effect of the introduction of configurable query threads in MySQL 8.0.23.

Now we make sure in such cases that `REORG_MOVED` is set before the checksum is calculated. (Bug #35783683)

- Following a node connection failure, the transporter registry's error state was not cleared before initiating a reconnect, which meant that the error causing the connection to be disconnected originally might still be set; this was interpreted as a failure to reconnect. (Bug #35774109)
- When encountering an **ENOMEM** (end of memory) error, the TCP transporter continued trying to send subsequent buffers which could result in corrupted data or checksum failures.

  We fix this by removing the **ENOMEM** handling from the TCP transporter, and waiting for sufficient memory to become available instead. (Bug #35700332)
- Setup of the binary log injector sometimes deadlocked with concurrent DDL. (Bug #35673915)
- The slow disconnection of a data node while a management server was unavailable could sometimes interfere with the rolling restart process. This became especially apparent when the cluster was hosted by NDB Operator, and the old `mgmd` pod did not recognize the IP address change of the restarted data node pod; this was visible as discrepancies in the output of `SHOW STATUS` on different management nodes.

  We fix this by making sure to clear any cached address when connecting to a data node so that the data node's new address (if any) is used instead. (Bug #35667611)
- The maximum permissible value for the oldest restorable global checkpoint ID is `MAX_INT32` (4294967295). Such an ID greater than this value causes the data node to shut down, requiring a backup and restore on a cluster started with `--initial`.

  Now, approximately 90 days before this limit is reached under normal usage, an appropriate warning is issued, allowing time to plan the required corrective action. (Bug #35641420)

  References: See also: Bug #35749589.
- Transactions whose size exceeded `binlog_cache_size` caused duplicate warnings. (Bug #35441583)
- NDB Cluster installation packages contained two copies of the `INFO_SRC` file. (Bug #35400142)
- Table map entries for some tables were written in the binary log, even though `log_replica_updates` was set to `OFF`. (Bug #35199996)
- The `NDB` source code is now formatted according to the rules used by `clang-format`, which it aligns it in this regard with the rest of the MySQL sources. (Bug #33517923)
- Subscription reports were sent out too early by `SUMA` during a node restart, which could lead to schema inconsistencies between cluster SQL nodes. In addition, an issue with the `ndbinfo restart_info` table meant that restart phases for nodes that did not belong to any node group were not always reported correctly. (Bug #30930132)
- Online table reorganization inserts rows from existing table fragments into new table fragments; then, after committing the inserted rows, it deletes the original rows. It was found that the inserts caused `SUMA` triggers to fire, and binary logging to occur, which led to the following issues:
  - Inconsistent behavior, since DDL is generally logged as one or more statements, if at all, rather than by row-level effect.
  - It was incorrect, since only writes were logged, but not deletes.
  - It was unsafe since tables with blobs did not receive associated the row changes required to form valid binary log events.
  - It used CPU and other resources needlessly.

For tables with no blob columns, this was primarily a performance issue; for tables having blob columns, it was possible for this behavior to result in unplanned shutdowns of mysqld processes performing binary logging and perhaps even data corruption downstream. (Bug #19912988)

References: See also: Bug #16028096, Bug #34843617.

- NDB API events are buffered to match the rates of production and consumption by user code. When the maximum size set to avoid unbounded memory usage when the rate is mismatched for an extended time was reached, event buffering stopped until the buffer usage dropped below a lower threshold; this manifested as an inability to find the container for latest epoch in when handling `NODE_FAILREP` events. To fix this problem, we add a `TE_OUT_OF_MEMORY` event to the buffer to inform the consumer that there may be missing events.

Changes in MySQL NDB Cluster 8.2.0 (2023-10-25)

Bugs Fixed

- **NDB Replication:** Updates to primary keys of character types were not correctly represented in the `BEFORE` and `AFTER` trigger values sent to the NDB binary log injector. This issue was previously fixed in part, but it was discovered subsequently that the problem still occurred when the `mysqld` was run with the binary logging options having the values listed here:

  - `--ndb-log-update-minimal=ON`
  - `--ndb-log-update-as-write=OFF`

  The minimal binary log format excluded all primary key columns from the `AFTER` values reflecting the updated row, the rationale for this being a flawed assumption that the primary key remained constant when an update trigger was received. This did not take into account the fact that, if the primary key uses a character data type, an update trigger is received if character columns are updated to values treated as equal by the comparison rules of the collation used.

  To be able to replicate such changes, we need to include them in the `AFTER` values; this fix ensures that we do so. (Bug #34540016)

  References: See also: Bug #27522732, Bug #34312769, Bug #34388068.

- **NDB Cluster APIs:** The header files `ndb_version.h` and `mgmapi.h` required C++ to compile, even though they should require C only. (Bug #35709497)

- **NDB Cluster APIs:** The MGM API functions `ndb_mgm_get_status()`, `ndb_mgm_get_status2()`, and `ndb_mgm_get_status3()` incorrectly returned `Illegal node status` on `Authorization failed` errors. (Bug #35687497)

- **NDB Cluster APIs:** `Ndb::pollEvents2()` did not set **`NDB_FAILURE_GCI`** (`~(Uint64)0`) to indicate cluster failure. (Bug #35671818)

  References: See also: Bug #31926584. This issue is a regression of: Bug #18753887.

- **NDB Client Programs:** When `ndb_select_all` failed to read all data from the table, it always tried to re-read it. This could lead to the two problems listed here:

  - Returning a non-empty partial result eventually led to spurious reports of duplicate rows.
  - The table header was printed on every retry.

  Now when `ndb_select_all` is unsuccessful at reading all the table data, its behavior is as follows:

  - When the result is non-empty, `ndb_select_all` halts with an error (and does not retry the scan of the table).

  - When the result is empty, `ndb_select_all` retries the scan, reusing the old header.

  (Bug #35510814)

- The bundled `certifi` library was updated to version 2024.2.2. (Bug #36618029)
- NDB Cluster did not compile using Clang 15. (Bug #35763112)
- A `DBUG_ENTER` in `src/common/transporter/Transporter.cpp` lacked a matching `DBUG_RETURN`. (Bug #35717730)
- When a `TransporterRegistry` (TR) instance connects to a management server, it first uses the MGM API, and then converts the connection to a `Transporter` connection for further communication. The initial connection had an excessively long timeout (60 seconds) so that, in the case of a cluster having two management servers where one was unavailable, clients were forced to wait until this management server timed out before being able to connect to the available one.

  We fix this by setting the MGM API connection timeout to 5000 milliseconds, which is equal to the timeout used by the TR for getting and setting dynamic ports. (Bug #35714466)
- Values for causes of conflicts used in conflict resolution exceptions tables were misaligned such that the order of `ROW_ALREADY_EXISTS` and `ROW_DOES_NOT_EXIST` was reversed. (Bug #35708719)
- When TLS is used over the TCP transporter, the `ssl_writev()` method may return `TLS_BUSY_TRY_AGAIN` in cases where the underlying `SSL_write()` returned either **`SSL_ERROR_WANT_READ`** or **`SSL_ERROR_WANT_WRITE`**, which is used to indicate to the upper layers that it is necessary to try the write again later.

  Since `TCP_Transporter::doSend()` may write in a loop in which multiple blocks of buffered data are written using a sequence of `writev()` calls, we may have successfully written some buffered data before encountering an **`SSL_ERROR_WANT_WRITE`**. In such cases the handling of the `TLS_BUSY_TRY_AGAIN` was simply to return from the loop, without first calling `iovec_data_sent(sum_sent)` in order to inform the buffering layer of what was sent.

  This resulted in later tries to resend a chunk which had already been sent, calling `writev()` with both duplicated data and an incorrect length argument. This resulted in a combination of checksum errors and SSL `writev()` failing with `bad length` errors reported in the logs.

  We fix this by breaking out of the send loop rather than just returning, so that execution falls through to the point in the code where such status updates are supposed to take place. (Bug #35693207)
- Removed a memory leak found in `src/mgmclient/main.cpp`. (Bug #35641639)
- When `DUMP 9993` was used in an attempt to release a signal block from a data node where a block had not been set previously using `DUMP 9992`, the data node shut down unexpectedly. (Bug #35619947)
- Improved `NDBFS` debugging output for bad requests. (Bug #35500304)

  References: This issue is a regression of: Bug #28922609.
- When other events led to `NDBFS` dumping requests to the log, some of the names of the request types were printed as `Unknown action`. (Bug #35499931)
- `ndb_restore` did not update compare-as-equal primary key values changed during backup. (Bug #35420131)
- Backups using `NOWAIT` did not start following a restart of the data node. (Bug #35389533)
- In cases where the distributed global checkpoint (GCP) protocol stops making progress, this is detected and optionally handled by the GCP monitor, with handling as determined by the

`TimeBetweenEpochsTimeout` and `TimeBetweenGlobalCheckpointsTimeout` data node parameters.

The LCP protocol is mostly node-local, but depends on the progress of the GCP protocol at the end of a local checkpoint (LCP); this means that, if the GCP protocol stalls, LCPs may also stall in this state. If the LCP watchdog detects that the LCP is stalled in this end state, it should defer to the GCP monitor to handle this situation, since the GCP Monitor is distribution-aware.

If no GCP monitor limit is set (`TimeBetweenEpochsTimeout` is equal 0), no handling of GCP stalls is performed by the GCP monitor. In this case, the LCP watchdog was still taking action which could eventually lead to cluster failure; this fix corrects this misbehavior so that the LCP watchdog no longer takes any such action. (Bug #29885899)

- Previously, when a timeout was detected during transaction commit and completion, the transaction coordinator (TC) switched to a serial commit-complete execution protocol, which slowed commit-complete processing for large transactions, affecting `GCP_COMMIT` delays and epoch sizes. Instead of switching in such cases, the TC now continues waiting for parallel commit-complete, periodically logging a transaction summary, with states and nodes involved. (Bug #22602898)

  References: See also: Bug #35260944.

Changes in MySQL NDB Cluster 8.1.0 (2023-07-18)

- IPv6 Support
- Functionality Added or Changed
- Bugs Fixed

IPv6 Support

- `NDB` did not start if IPv6 support was not enabled on the host, even when no nodes in the cluster used any IPv6 addresses. (Bug #106485, Bug #33324817, Bug #33870642, WL #15661)

Functionality Added or Changed

- **Important Change; NDB Cluster APIs:** The `NdbRecord` interface allows equal changes of primary key values; that is, you can update a primary key value to its current value, or to a value which compares as equal according to the collation rules being used, without raising an error. `NdbRecord` does not itself try to prevent the update; instead, the data nodes check whether a primary key is updated to an unequal value and in this case reject the update with Error 897: `Update attempt of primary key via ndbcluster internal api`.

  Previously, when using any other mechanism than `NdbRecord` in an attempt to update a primary key value, the NDB API returned error 4202 `Set value on tuple key attribute is not allowed`, even setting a value identical to the existing one. With this release, the check when performing updates by other means is now passed off to the data nodes, as it is already by `NdbRecord`.

  This change applies to performing primary key updates with `NdbOperation::setValue()`, `NdbInterpretedCode::write_attr()`, and other methods of these two classes which set column values (including `NdbOperation` methods `incValue()`, `subValue()`, `NdbInterpretedCode` methods `add_val()`, `sub_val()`, and so on), as well as the `OperationOptions::OO_SETVALUE` extension to the `NdbOperation` interface. (Bug #35106292)

Bugs Fixed

- **NDB Cluster APIs:** Printing of debug log messages was enabled by default for `Ndb_cluster_connection`. (Bug #35416908)

  References: See also: Bug #35927.

- **NDB Cluster APIs:** While setting up an `NdbEventOperation`, it is possible to pass a pointer to a buffer provided by the application; when data is later received, it should be available in that specified location.

  The received data was properly placed in the provided buffer location, but the NDB API also allocated internal buffers which, subsequently, were not actually needed, ultimately wasting resources. This problem primarily manifested itself in applications subscribing to data changes from `NDB` using the `NdbEventOperation::getValue()` and `getPreValue()` functions with the buffer provided by application.

  To remedy this issue, we no longer allocate internal buffers in such cases. (Bug #35292716)

- When dropping an `NdbEventOperation` after use, the `ndbcluster` plugin now first explicitly clears the object's custom data area. (Bug #35424845)

- After a socket polled as readable in `NdbSocket::readln()`, it was possible for `SSL_peek()` to block in the kernel when the TLS layer held no application data. We fix this by releasing the lock on the user mutex during `SSL_peek()`, as well as when polling. (Bug #35407354)

- When handling the connection (or reconnection) of an API node, it was possible for data nodes to inform the API node that it was permitted to send requests too quickly, which could result in requests not being delivered and subsequently timing out on the API node with errors such as Error 4008 `Receive from Ndb failed` or Error 4012 `Request ndbd time-out, maybe due to high load or communication problems`. (Bug #35387076)

- Made the following improvements in warning output:

  - Now, in addition to local checkpoint (LCP) elapsed time, the maximum time allowed without any progress is also printed.

  - Table IDs and fragment IDs are undefined and thus not relevant when an LCP has reached `WAIT_END_LCP` state, and are no longer printed at that point.

  - When the maximum limit was reached, the same information was shown twice, as both warning and crash information.

  (Bug #35376705)

- Memory consumption of long-lived threads running inside the `ndbcluster` plugin grew when accessing the data dictionary. (Bug #35362906)

- A failure to connect could lead `ndb_restore` to exit with code 1, without reporting any error message. Now we supply an appropriate error message in such cases. (Bug #35306351)

- When deferred triggers remained pending for an uncommitted transaction, a subsequent transaction could waste resources performing unnecessary checks for deferred triggers; this could lead to an unplanned shutdown of the data node if the latter transaction had no committable operations.

  This was because, in some cases, the control state was not reinitialized for management objects used by `DBTC`.

  We fix this by making sure that state initialization is performed for any such object before it is used. (Bug #35256375)

- A pushdown join between queries featuring very large and possibly overlapping `IN()` and `NOT IN()` lists caused SQL nodes to exit unexpectedly. One or more of the `IN()` (or `NOT IN()`) operators required in excess of 2500 arguments to trigger this issue. (Bug #35185670, Bug #35293781)

- The buffers allocated for a key of size `MAX_KEY_SIZE` were of insufficient size. (Bug #35155005)

- The fix for a previous issue added a check to ensure that fragmented signals are never sent to `V_QUERY` blocks, but this check did not take into account that, when the receiving node is not a data node, the block number is not applicable. (Bug #35154637)

  References: This issue is a regression of: Bug #34776970.

- `ndbcluster` plugin log messages now use `SYSTEM` as the log level and `NDB` as the subsystem for logging. This means that informational messages from the `ndbcluster` plugin are always printed; their verbosity can be controlled by using `--ndb_extra_logging`. (Bug #35150213)

- We no longer print an informational message `Validating excluded objects` to the SQL node's error log every `ndb_metadata_check_interval` seconds (default 60) when `log_error_verbosity` is greater than or equal to 3 (`INFO` level). It was found that such messages flooded the error log, making it difficult to examine and using excess disk space, while not providing any additional benefit. (Bug #35103991)

- Some calls made by the `ndbcluster` handler to `push_warning_printf()` used severity level `ERROR`, which caused an assertion in debug builds. This fix changes all such calls to use severity `WARNING` instead. (Bug #35092279)

- When a connection between a data node and an API or management node was established but communication was available only from the other node to the data node, the data node considered the other node “live”, since it was receiving heartbeats, but the other node did not monitor heartbeats and so reported no problems with the connection. This meant that the data node assumed wrongly that the other node was (fully) connected.

  We solve this issue by having the API or management node side begin to monitor data node liveness even before receiving the first `REGCONF` signal from it; the other node sends a `REGREQ` signal every 100 milliseconds, and only if it receives no `REGCONF` from the data node in response within 60 seconds is the node reported as disconnected. (Bug #35031303)

- The data node process printed a stack trace during program exit due to conditions other than software errors, leading to possible confusion in some cases. (Bug #34836463)

  References: See also: Bug #34629622.

- The log contained a high volume of messages having the form `DICT: index index number stats auto-update requested`, logged by the `DBDICT` block each time it received a report from `DBTUX` requesting an update. These requests often occur in quick succession during writes to the table, with the additional possibility in this case that duplicate requests for updates to the same index were being logged.

  Now we log such messages just before `DBDICT` actually performs the calculation. This removes duplicate messages and spaces out messages related to different indexes. Additional debug log messages are also introduced by this fix, to improve visibility of the decisions taken and calculations performed. (Bug #34760437)

- A comparison check in `Dblqh::handle_nr_copy()` for the case where two keys were not binary-identical could still compare as equal by collation rules if the key had any character columns, but did not actually check for the existence of the keys. This meant it was possible to call `xfrm_key()` with an undefined key. (Bug #34734627)

  References: See also: Bug #34681439. This issue is a regression of: Bug #30884622.

- When a data node process received a Unix signal (such as with `kill -6`), the signal handler function showed a stack trace, then called `ErrorReporter`, which also showed a stack trace. Now in such cases, `ErrorReporter` checks for this situation and does not print a stack trace of its own when called from the signal handler. (Bug #34629622)

  References: See also: Bug #34836463.

- Local checkpoints (LCPs) wait for a global checkpoint (GCP) to finish for a fixed time during the end phase, so they were performed sometimes even before all nodes were started.

  In addition, this bound, calculated by the GCP coordinator, was available only on the coordinator itself, and only when the node had been started (start phase 101).

  These two issues are fixed by calculating the bound earlier in start phase 4; GCP participants also calculate the bound whenever a node joins or leaves the cluster. (Bug #32528899)

- When an `ALTER TABLE` adds columns to a table, the `maxRecordSize` used by local checkpoints to allocate buffer space for rows may change; this is set in a `GET_TABINFOCONF` signal and used again later in `BACKUP_FRAGMENT_REQ`. If, during the gap between these two signals, an `ALTER TABLE` changed the number of columns, the value of `maxRecordSize` used could be stale, thus be inaccurate, and so lead to further issues.

  Now we always update `maxRecordSize` (from `DBTUP`) on receipt of a `BACKUP_FRAGMENT_REQ` signal, before attempting the allocation of the row buffer. (Bug #105895, Bug #33680100)

Index

A

ALTER TABLE, 43
API nodes, 36, 43
API timeouts, 10, 27
ApiConnectRecord, 10, 27
ApiFailureHandlingTimeout, 7, 25
AutomaticThreadConfig, 15, 31

B

backports, 36
backup, 7, 25
BINARY, 15, 31
binary log, 15
binary log injector, 18, 33, 36
binary log rotation, 10
binary logging, 36
bind address, 18, 33
binlog_cache_size, 36
BIT, 18
BLOB_INLINE_SIZE, 10, 13, 27, 29
blocking, 43

C

CA-days, 15, 31
certifi, 41
certificates, 15, 31
changes
  NDB Cluster, 22
checkpoints, 43
Clang, 41
Cleaner, 7, 25
comparisons, 43
compile-cluster, 7, 25
compiling, 15, 31, 36, 41
condition pushdown, 15, 31
configuration, 13, 29
conflict resolution, 41
connection timeouts, 41

connections, 43
const, 15, 31
CopyFrag, 15, 31
CopyGCI, 10, 27
CREATE TABLESPACE, 6, 24

D

data dictionary, 43
data nodes, 43
DBLQH, 15, 31
DBSPJ, 10, 27, 36
DBTC, 10, 15, 27, 31, 36, 43
disable-indexes, 18, 33
DUMP commands, 41

E

encryption, 6, 24
ENOMEM, 36
error handling, 18, 33
errors, 6, 24, 36, 43
event buffer, 36
EventBuffer, 18, 33
events, 36
expat, 18

F

failure handling, 13, 29
free memory, 18, 33

G

GCI, 36
GCP, 41
GCP stalls, 41
global schema lock, 15, 18, 31, 33
grants, 36

H

hash scan, 18
heartbeats, 18, 33
help text, 18, 33

I

Important Change, 15, 31, 43
IN(), 43
include-stored-grants, 7, 25
index statistics, 13, 29
INFORMATION_SCHEMA, 18
INPLACE, 15, 31
InvocationHandler, 7, 25
IPv6 support, 43

J

joins, 43

L

language support, 36

libssh, 15
libxml2, 7, 15, 25
Linux, 18, 33
lld, 15, 31
log level, 5, 23
logging, 6, 10, 15, 24, 27, 31, 36, 43
logs, 18, 33
log_replica, 18
log_replica_updates, 36
LQH_TRANSCONF, 7, 25

M

macOS, 10, 27
maxRecordSize, 43
metadata lock, 36
Microsoft Windows, 10, 27
MySQL NDB ClusterJ, 7, 13, 25, 29
mysql.ndb_apply_status table, 15
mysqld, 18, 33

N

NDB Client Programs, 6, 7, 13, 15, 18, 24, 25, 29, 31, 33, 41
NDB Cluster, 4, 5, 6, 7, 10, 13, 15, 18, 22, 23, 24, 25, 27, 29, 31, 33, 36, 41, 43
NDB Cluster APIs, 10, 13, 15, 18, 27, 29, 31, 33, 36, 41, 43
NDB Disk Data, 10, 27
NDB Operator, 36
NDB Replication, 7, 10, 15, 18, 36, 41
ndb-wait-setup, 18, 33
Ndb::pollEvents2(), 41
ndbd, 15, 31
NdbDictionaryImpl.cpp, 15, 31
NdbEventOperation, 18, 33, 43
NDBFS, 41
ndbinfo Information Database, 7, 10, 15, 18, 25, 27, 31, 33, 36
ndbinfo.restart_info, 36
NdbInterpretedCode, 43
NdbOperation, 43
NdbRecord, 13, 29
Ndb_cluster_connection, 10, 27, 43
ndb_config, 10, 27
ndb_log_apply_status, 15
ndb_log_cache_size, 13, 29
ndb_log_update_as_write, 41
ndb_log_update_minimal, 41
ndb_metadata_check, 7, 25
ndb_metadata_check_interval, 43
ndb_mgmd, 18, 33
ndb_mgm_get_status(), 41
ndb_mgm_get_status2(), 41
ndb_mgm_get_status3(), 41
ndb_print_backup_file, 36
ndb_redo_log_reader, 15, 18, 31, 33
ndb_restore, 15, 31, 41, 43
ndb_schema_object, 7, 25
NDB_SCHEMA_OBJECT, 15, 31
Ndb_schema_participant_count, 7, 25

ndb_select_all, 41
ndb_sign_keys, 15, 18, 31, 33, 36
ndb_size.pl, 13, 29
NDB_STORED_USER, 36
ndb_waiter, 18, 33
neighbor nodes, 18, 33
node IDs, 10, 27
node shutdown, 13, 29
NOWAIT, 41
NumCPUs, 15, 31

O

ODirect, 15, 31
online operations, 15, 31
online table reorganization, 36
OOBR, 7, 25
OpenJPA, 13, 29
openSSL, 15, 31
out of buffer errors, 7, 25

P

Packaging, 18, 33
packaging, 36
primary key updates, 41
primary keys, 41, 43
PURGE BINARY LOGS, 7, 25

Q

QMGR, 7, 25, 36

R

readln_socket(), 36
read_only, 36
REGCONF, 43
REGREQ, 43
releaseGlobal(), 36
REORG_MOVED, 36
REPORT BACKUPSTATUS, 7, 25
restarts, 15, 31
rolling restart, 36

S

scans, 10, 27
schema distribution, 18, 33
send buffer, 18
send threads, 18, 33
SendBuffer, 18, 33
SHM, 13, 29
signal dump, 7, 25
sockets, 18, 33, 36
source code, 36
SSL, 18, 33
ssl_write(), 36
STMT_END_F, 10
STOP, 18, 33
system restarts, 10, 27

T

TABLESPACES table, 18
TC, 41
TCRELEASEREQ, 10, 27
testsuite, 7, 25
TLS, 7, 15, 18, 25, 31, 33, 36, 41
transaction coordinator, 15, 31
transactions, 18, 33
Transporter.cpp, 41
TransporterRegistry, 41
transporters, 18, 33, 36
transporter_details, 15, 18, 31, 33
transporter_details table, 18
TRIX, 13, 29
TRPMAN, 18, 33

U

Ubuntu, 36
upgrades, 36

V

V_QUERY, 43

W

warnings, 18, 33, 43
watchdog, 7, 25