# 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.

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 legal information, see the Legal Notices.

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

Document generated on: 2024-04-19 (revision: 28245)

## Table of Contents



## 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 8.4.0 (2024-04-30, LTS Release)

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, LTS Release)).

- Deprecation and Removal Notes

## 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)

## Functionality Added or Changed

- **Packaging:** Added support for Fedora 40.
- **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:** 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:** 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)

- `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 NDB Cluster 8.4, and includes changes for all NDB Cluster releases since NDB 8.0.

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 8.4.0 (2024-04-30, LTS Release)

- 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

- 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:** 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:** 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)

- 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)

- `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, Innovation Release)



## 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 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.

- 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)
- 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, Innovation Release)

## Bugs Fixed

- **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 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)
- 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, Innovation Release)



### 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

## P



## Q



## R



## S



## T



## U



## V



## W