MySQL NDB Cluster 8.2 Release Notes

Abstract

This document contains release notes for the changes in MySQL NDB Cluster version 8.2.

NDB Cluster 8.2 is based on MySQL Server 8.2 and uses version 8.2 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 8.2.

For general information about features added in NDB Cluster 8.2, see What is New in MySQL NDB Cluster 8.2. 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 8.2 documentation, see the MySQL 8.2 Reference Manual, which includes an overview of features added in MySQL 8.2 that are not specific to NDB Cluster (What Is New in MySQL 8.2), and discussion of upgrade issues that you may encounter for upgrades from MySQL 8.1 to MySQL 8.2 (Changes in MySQL 8.2). For a complete list of all bug fixes and feature changes made in MySQL 8.2 that are not specific to `NDB`, see Changes in MySQL 8.2.0 (2023-10-25, Innovation Release).

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: 2023-11-28 (revision: 27604)

Table of Contents

Preface and Legal Notices

This document contains release notes for MySQL NDB Cluster version 8.2.

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 8.2.0 (2023-10-25, Innovation Release)

MySQL NDB Cluster 8.2 is an Innovation release of NDB 8.2, based on MySQL Server 8.2 and including features in version 8.2 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.2. NDB Cluster 8.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.2, see What is New in MySQL NDB Cluster 8.2.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes made in mainline MySQL 8.2 (see Changes in MySQL 8.2.0 (2023-10-25, Innovation Release)).

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)

- 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.