MySQL NDB Cluster 8.0 Release Notes

Abstract

This document contains release notes for the changes in each release of MySQL NDB Cluster that uses version 8.0 of the `NDB` (`NDBCLUSTER`) storage engine.

Each NDB Cluster 8.0 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 8.0.

For general information about features added in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0. 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 8.0 documentation, see the MySQL 8.0 Reference Manual, which includes an overview of features added in MySQL 8.0 that are not specific to NDB Cluster (What Is New in MySQL 8.0), and discussion of upgrade issues that you may encounter for upgrades from MySQL 5.6 to MySQL 8.0 (Changes in MySQL 8.0). For a complete list of all bug fixes and feature changes made in MySQL 8.0 that are not specific to `NDB`, see MySQL 8.0 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: 2025-07-02 (revision: 30204)

Table of Contents

Preface and Legal Notices 2
Changes in MySQL NDB Cluster 8.0.43 (Not yet released, General Availability) 4
Changes in MySQL NDB Cluster 8.0.42 (2025-04-16, General Availability) 4
Changes in MySQL NDB Cluster 8.0.41 (2025-01-22, General Availability) 7
Changes in MySQL NDB Cluster 8.0.40 (2024-10-16, General Availability) 10
Changes in MySQL NDB Cluster 8.0.39 (2024-07-23, General Availability) 11
Changes in MySQL NDB Cluster 8.0.38 (2024-07-02, General Availability) 12
Changes in MySQL NDB Cluster 8.0.37 (2024-04-30, General Availability) 14
Changes in MySQL NDB Cluster 8.0.36 (2024-01-16, General Availability) 17
Changes in MySQL NDB Cluster 8.0.35 (2023-10-25, General Availability) 21
Changes in MySQL NDB Cluster 8.0.34 (2023-07-18, General Availability) 24
Changes in MySQL NDB Cluster 8.0.33 (2023-04-18, General Availability) 27
Changes in MySQL NDB Cluster 8.0.32 (2023-01-17, General Availability) 33
Changes in MySQL NDB Cluster 8.0.31 (2022-10-11, General Availability) 37
Changes in MySQL NDB Cluster 8.0.30 (2022-07-26, General Availability) 42
Changes in MySQL NDB Cluster 8.0.29 (2022-04-26, General Availability) 48
Changes in MySQL NDB Cluster 8.0.28 (2022-01-18, General Availability) 56
Changes in MySQL NDB Cluster 8.0.27 (2021-10-19, General Availability) 61
Changes in MySQL NDB Cluster 8.0.26 (2021-07-20, General Availability) 67
Changes in MySQL NDB Cluster 8.0.25 (2021-05-11, General Availability) 72

Changes in MySQL NDB Cluster 8.0.24 (2021-04-20, General Availability) 73
Changes in MySQL NDB Cluster 8.0.23 (2021-01-18, General Availability) 80
Changes in MySQL NDB Cluster 8.0.22 (2020-10-19, General Availability) 86
Changes in MySQL NDB Cluster 8.0.21 (2020-07-13, General Availability) 94
Changes in MySQL NDB Cluster 8.0.20 (2020-04-27, General Availability) 99
Changes in MySQL NDB Cluster 8.0.19 (2020-01-13, General Availability) 104
Changes in MySQL NDB Cluster 8.0.18 (2019-10-14, Release Candidate) 112
Changes in MySQL NDB Cluster 8.0.17 (2019-07-22, Release Candidate) 119
Changes in MySQL NDB Cluster 8.0.16 (2019-04-25, Development Milestone) 124
Changes in MySQL NDB Cluster 8.0.15 (Not released) 132
Changes in MySQL NDB Cluster 8.0.14 (2019-01-21, Development Milestone) 133
Changes in MySQL NDB Cluster 8.0.13 (2018-10-23, Development Milestone) 136
Release Series Changelogs: MySQL NDB Cluster 8.0 145
Changes in MySQL NDB Cluster 8.0.38 (2024-07-02, General Availability) 145
Changes in MySQL NDB Cluster 8.0.37 (2024-04-30, General Availability) 148
Changes in MySQL NDB Cluster 8.0.36 (2024-01-16, General Availability) 150
Changes in MySQL NDB Cluster 8.0.35 (2023-10-25, General Availability) 153
Changes in MySQL NDB Cluster 8.0.34 (2023-07-18, General Availability) 155
Changes in MySQL NDB Cluster 8.0.33 (2023-04-18, General Availability) 158
Changes in MySQL NDB Cluster 8.0.32 (2023-01-17, General Availability) 163
Changes in MySQL NDB Cluster 8.0.31 (2022-10-11, General Availability) 167
Changes in MySQL NDB Cluster 8.0.30 (2022-07-26, General Availability) 172
Changes in MySQL NDB Cluster 8.0.29 (2022-04-26, General Availability) 177
Changes in MySQL NDB Cluster 8.0.28 (2022-01-18, General Availability) 185
Changes in MySQL NDB Cluster 8.0.27 (2021-10-19, General Availability) 188
Changes in MySQL NDB Cluster 8.0.26 (2021-07-20, General Availability) 194
Changes in MySQL NDB Cluster 8.0.24 (2021-04-20, General Availability) 199
Changes in MySQL NDB Cluster 8.0.23 (2021-01-18, General Availability) 206
Changes in MySQL NDB Cluster 8.0.22 (2020-10-19, General Availability) 212
Changes in MySQL NDB Cluster 8.0.21 (2020-07-13, General Availability) 219
Changes in MySQL NDB Cluster 8.0.20 (2020-04-27, General Availability) 224
Changes in MySQL NDB Cluster 8.0.19 (2020-01-13, General Availability) 229
Changes in MySQL NDB Cluster 8.0.18 (2019-10-14, Release Candidate) 236
Changes in MySQL NDB Cluster 8.0.17 (2019-07-22, Release Candidate) 243
Changes in MySQL NDB Cluster 8.0.16 (2019-04-25, Development Milestone) 247
Changes in MySQL NDB Cluster 8.0.14 (2019-01-21, Development Milestone) 256
Changes in MySQL NDB Cluster 8.0.13 (2018-10-23, Development Milestone) 259
Index 268

Preface and Legal Notices

This document contains release notes for the changes in each release of MySQL NDB Cluster that uses version 8.0 of the `NDB` storage engine.

Legal Notices

Copyright © 1997, 2025, Oracle and/or its affiliates.

License Restrictions

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

Warranty Disclaimer

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

Restricted Rights Notice

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

Hazardous Applications Notice

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Trademark Notice

Oracle, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

Third-Party Content, Products, and Services Disclaimer

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Use of This Documentation

This documentation is NOT distributed under a GPL license. Use of this documentation is subject to the following terms:

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the

documentation in a manner similar to how Oracle disseminates it (that is, electronically for download on a Web site with the software) or on a CD-ROM or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Oracle. Oracle and/or its affiliates reserve any and all rights to this documentation not expressly granted above.

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 8.0.43 (Not yet released, General Availability)

MySQL NDB Cluster 8.0.43 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.43 (see Changes in MySQL 8.0.43 (Not yet released, General Availability)).

Version 8.0.43-ndb-8.0.43 has no release notes, or they have not been published because the product version has not been released.

Changes in MySQL NDB Cluster 8.0.42 (2025-04-16, General Availability)

MySQL NDB Cluster 8.0.42 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.42 (see Changes in MySQL 8.0.42 (2025-04-15, General Availability)).

- 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 #37535513)

- **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)

- 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 transcations, 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, refering 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)
- 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)

- 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)
- 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 8.0.41 (2025-01-22, General Availability)

MySQL NDB Cluster 8.0.41 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.41 (see Changes in MySQL 8.0.41 (2025-01-21, General Availability)).

- 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

- **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 sersus 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 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.

- 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 8.0.40 (2024-10-16, General Availability)

MySQL NDB Cluster 8.0.40 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.40 (see Changes in MySQL 8.0.40 (2024-10-15, General Availability)).

- 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)

- The `AES_ENCRYPT()` function did not always return a valid result. (Bug #37039383)
- 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 8.0.39 (2024-07-23, General Availability)

MySQL NDB Cluster 8.0.39 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.39 (see Changes in MySQL 8.0.39 (2024-07-23, General Availability)).

This release contains no functional changes specific to MySQL NDB Cluster, and is published to align with and include changes made in MySQL Server 8.0.39.

Changes in MySQL NDB Cluster 8.0.38 (2024-07-02, General Availability)

MySQL NDB Cluster 8.0.38 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.38 (see Changes in MySQL 8.0.38 (2024-07-01, General Availability)).

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 8.0.39 instead.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **ndbinfo Information Database:** The following columns have been added to the `transporter_details` table:
  - `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.
  - `type`: The connection type used by this transporter (`TCP` or `SHM`).

  See The ndbinfo transporter_details Table, for more information. (Bug #36579842)

  References: See also: Bug #36569947.

- **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 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 from `src/ndbapi/NdbDictionaryImpl.cpp`. (Bug #36532102)

- 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)

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

- An error injected in the `LQH` proxy and then resent to all workers was eventually cleared by the workers, but not in the `LQH` proxy where it was never cleared. (Bug #36398772)
- 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)

Changes in MySQL NDB Cluster 8.0.37 (2024-04-30, General Availability)

MySQL NDB Cluster 8.0.37 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.37 (see Changes in MySQL 8.0.37 (2024-04-30, General Availability)).

- 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.
- **Packaging:** Added support for Fedora 40 and Ubuntu 24.04.
- **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 Client Programs:** `ndb_redo_log_reader` could not read data from encrypted files. (Bug #36313482)
- **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:** `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:** Invoking ndb_mgmd with the --bind-address option could in some cases cause the program to terminate unexpectedly. (Bug #36263410)
- **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 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)
- The bundled `certifi` library was updated to version 2024.2.2. (Bug #36618029)
- 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)

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

- 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)
- 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.
- 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 8.0.36 (2024-01-16, General Availability)

MySQL NDB Cluster 8.0.36 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.36 (see Changes in MySQL 8.0.36 (2024-01-16, General Availability)).

- Compilation Notes
- Bugs Fixed

Compilation Notes

- 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)

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:** 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 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)

- **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)

- 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.

- 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)
- 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.
- 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)
- 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)
- 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)
- 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 8.0.35 (2023-10-25, General Availability)

MySQL NDB Cluster 8.0.35 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.35 (see Changes in MySQL 8.0.35 (2023-10-25, General Availability)).

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:** `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)

- 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)

- 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)

- 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.

- 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.

- 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.

- 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)

Changes in MySQL NDB Cluster 8.0.34 (2023-07-18, General Availability)

MySQL NDB Cluster 8.0.34 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.34 (see Changes in MySQL 8.0.34 (2023-07-18, General Availability)).

- 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 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.
- 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)

Changes in MySQL NDB Cluster 8.0.33 (2023-04-18, General Availability)

MySQL NDB Cluster 8.0.33 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.33 (see Changes in MySQL 8.0.33 (2023-04-18, General Availability)).

- Parallel Event Execution (Multithreaded Replica)
- Functionality Added or Changed
- Bugs Fixed

Parallel Event Execution (Multithreaded Replica)

- **NDB Replication:** NDB Cluster replication now supports the MySQL multithreaded applier (MTA) on replica servers. This makes it possible for binary log transactions to be applied in parallel on the replica, increasing peak replication throughput. To enable this on the replica, it is necessary to perform the following steps:

  - Set the `--ndb-log-transaction-dependency` option, added in this release, to `ON`. This must be done on startup of the source `mysqld`.
  - Set the `binlog_transaction_dependency_tracking` server system variable to `WRITESET`, also on the source, which causes transaction dependencies to be determined at the source. This can be done at runtime.
  - Make sure the replica uses multiple worker threads; this is determined by the value of the `replica_parallel_workers` server system variable, which `NDB` now honors (previously, `NDB` effectively ignored any value set for this variable). The default is 4, and can be changed on the replica at runtime.

  You can adjust the size of the buffer used to store the transaction dependency history on the source using the `--binlog-transaction-dependency-history-size` option. The source should also have `replica_parallel_type` set to `LOGICAL_CLOCK` (the default).

  Additionally, on the replica, `replica_preserve_commit_order` must be `ON` (the default).

  For more information about the MySQL replication applier, see Replication Threads. For more information about NDB Cluster replication and the multithreaded applier, see NDB Cluster Replication Using the Multithreaded Applier. (Bug #27960, Bug #11746675, Bug #35164090, Bug #34229520, WL #14885, WL #15145, WL #15455, WL #15457)

Functionality Added or Changed

- **NDB Cluster APIs:** The `Node.js` library used to build the MySQL NoSQL Connector for JavaScript has been upgraded to version 18.12.1. (Bug #35095122)
- **MySQL NDB ClusterJ:** Performance has been improved for accessing tables using a single-column partition key when the column is of type `CHAR` or `VARCHAR`. (Bug #35027961)
- Beginning with this release, `ndb_restore` implements the `--timestamp-printouts` option, which causes all error, info, and debug node log messages to be prefixed with timestamps. (Bug #34110068)

Bugs Fixed

- **Microsoft Windows:** Two memory leaks found by code inspection were removed from `NDB` process handles on Windows platforms. (Bug #34872901)
- **Microsoft Windows:** On Windows platforms, the data node angel process did not detect whether a child data node process exited normally. We fix this by keeping an open process handle to the child and using this when probing for the child's exit. (Bug #34853213)
- **NDB Replication:** When using a multithreaded applier, the `start_pos` and `end_pos` columns of the `ndb.apply_status` table (see ndb_apply_status Table) did not contain the correct position information. (Bug #34806344)
- **NDB Cluster APIs; MySQL NDB ClusterJ:** MySQL ClusterJ uses a scratch buffer for primary key hash calculations which was limited to 10000 bytes, which proved too small in some cases. Now we `malloc()` the buffer if its size is not sufficient.

  This also fixes an issue with the `Ndb` object methods `startTransaction()` and `computeHash()` in the NDB API: Previously, if either of these methods was passed a temporary buffer of insufficient size, the method failed. Now in such cases a temporary buffer is allocated.

  Our thanks to Mikael Ronström for this contribution. (Bug #103814, Bug #32959894)
- **NDB Cluster APIs:** When dropping an event operation (`NdbEventOperation`) in the NDB API, it was sometimes possible for the dropped event operation to remain visible to the application after instructing the data nodes to stop sending events related to this event operation, but before all pending buffered events were consumed and discarded. This could be observed in certain cases when performing an online alter operation, such as `ADD COLUMN` or `RENAME COLUMN`, along with concurrent writes to the affected table.

  Further analysis showed that the dropped events were accessible when iterating through event operations with `Ndb::getGCIEventOperations()`. Now, this method skips dropped events when called iteratively. (Bug #34809944)
- **NDB Cluster APIs:** `Event::getReport()` always returned an error for an event opened from `NDB`, instead of returning the flags actually used by the report object. (Bug #34667384)
- Before a new `NDB` table definition can be stored in the data dictionary, any existing definition must be removed. Table definitions have two unique values, the table name and the NDB Cluster `se_private_id`. During installation of a new table definition, we check whether there is any existing definition with the same table name and, if so, remove it. Then we check whether the table removed and the one being installed have the same `se_private_id`; if they do not, any definition that is occupying this `se_private_id` is considered stale, and removed as well.

  Problems arose when no existing definition was found by the search using the table's name, since no definition was dropped even if one occupied `se_private_id`, leading to a duplicate key error when attempting to store the new table. The internal `store_table()` function attempted to clear the diagnostics area, remove the stale definition of `se_private_id`, and try to store it once again, but

the diagnostics area was not actually cleared, thus leaking the error is thus leaked and presenting it to the user.

To fix this, we remove any stale table definition, regardless of any action taken (or not) by `store_table()`. (Bug #35089015)

- Fixed the following two issues in the output of `ndb_restore`:
  - The backup file format version was shown for both the backup file format version and the version of the cluster which produced the backup.
  - To reduce confusion between the version of the file format and the version of the cluster which produced the backup, the backup file format version is now shown using hexadecimal notation.

  (Bug #35079426)

  References: This issue is a regression of: Bug #34110068.

- Removed a memory leak in the `DBDICT` kernel block caused when an internal foreign key definition record was not released when no longer needed. This could be triggered by either of the following events:
  - Drop of a foreign key constraint on an `NDB` table
  - Rejection of an attempt to create a foreign key constraint on an `NDB` table

  Such records use the `DISK_RECORDS` memory resource; you can check this on a running cluster by executing `SELECT node_id, used FROM ndbinfo.resources WHERE resource_name='DISK_RECORDS'` in the `mysql` client. This resource uses `SharedGlobalMemory`, exhaustion of which could lead not only to the rejection of attempts to create foreign keys, but of queries making use of joins as well, since the `DBSPJ` block also uses shared global memory by way of `QUERY_MEMORY`. (Bug #35064142)

- When attempting a copying alter operation with `--ndb-allow-copying-alter-table = OFF`, the reason for rejection of the statement was not always made clear to the user. (Bug #35059079)

- When a transaction coordinator is starting fragment scans with many fragments to scan, it may take a realtime break (RTB) during the process to ensure fair CPU access for other requests. When the requesting API disconnected and API failure handling for the scan state occurred before the RTB continuation returned, continuation processing could not proceed because the scan state had been removed.

  We fix this by adding appropriate checks on the scan state as part of the continuation process. (Bug #35037683)

- Sender and receiver signal IDs were printed in trace logs as signed values even though they are actually unsigned 32-bit numbers. This could result in confusion when the top bit was set, as it cuased such numbers to be shown as negatives, counting upwards from `-MAX_32_BIT_SIGNED_INT`. (Bug #35037396)

- A fiber used by the `DICT` block monitors all indexes, and triggers index statistics calculations if requested by `DBTUX` index fragment monitoring; these calculations are performed using a schema transaction. When the `DICT` fiber attempts but fails to seize a transaction handle for requesting a schema transaction to be started, fiber exited, so that no more automated index statistics updates could be performed without a node failure. (Bug #34992370)

  References: See also: Bug #34007422.

- Schema objects in NDB use composite versioning, comprising major and minor subversions. When a schema object is first created, its major and minor versions are set; when an existing schema object is altered in place, its minor subversion is incremented.

At restart time each data node checks schema objects as part of recovery; for foreign key objects, the versions of referenced parent and child tables (and indexes, for foreign key references not to or from a table's primary key) are checked for consistency. The table version of this check compares only major subversions, allowing tables to evolve, but the index version also compares minor subversions; this resulted in a failure at restart time when an index had been altered.

We fix this by comparing only major subversions for indexes in such cases. (Bug #34976028)

References: See also: Bug #21363253.

- `ndb_import` sometimes silently ignored hint failure for tables having large `VARCHAR` primary keys. For hinting which transaction coordinator to use, `ndb_import` can use the row's partitioning key, using a 4092 byte buffer to compute the hash for the key.

  This was problematic when the key included a `VARCHAR` column using UTF8, since the hash buffer may require in bytes up to 24 times the number of maximum characters in the column, depending on the column's collation; the hash computation failed but the calling code in `ndb_import` did not check for this, and continued using an undefined hash value which yielded an undefined hint.

  This did not lead to any functional problems, but was not optimal, and the user was not notified of it.

  We fix this by ensuring that `ndb_import` always uses sufficient buffer for handling character columns (regardless of their collations) in the key, and adding a check in `ndb_import` for any failures in hash computation and reporting these to the user. (Bug #34917498)

- When the `ndbcluster` plugin creates the `ndb_schema` table, the plugin inserts a row containing metadata, which is needed to keep track of this NDB Cluster instance, and which is stored as a set of key-value pairs in a row in this table.

  The `ndb_schema` table is hidden from MySQL and so not possible to query using SQL, but contains a UUID generated by the same MySQL server that creates the `ndb_schema` table; the same UUID is also stored as metadata in the data dictionary of each MySQL Server when the `ndb_schema` table is installed on it.

  When a `mysqld` connects (or reconnects) to `NDB`, it compares the UUID in its own data dictionary with the UUID stored in `NDB` in order to detect whether it is reconnecting to the same cluster; if not, the entire contents of the data dictionary are scrapped in order to make it faster and easier to install all tables fresh from `NDB`.

  One such case occurs when all `NDB` data nodes have been restarted with `--initial`, thus removing all data and tables. Another happens when the `ndb_schema` table has been restored from a backup without restoring any of its data, since this means that the row for the `ndb_schema` table would be missing.

  To deal with these types of situations, we now make sure that, when synchronization has completed, there is always a row in the `NDB` dictionary with a UUID matching the UUID stored in the MySQL server data dictionary. (Bug #34876468)

- When running an NDB Cluster with multiple management servers, termination of the `ndb_mgmd` processes required an excessive amount of time when shutting down the cluster. (Bug #34872372)

- Schema distribution timeout was detected by the schema distribution coordinator after dropping and re-creating the `mysql.ndb_schema` table when any nodes that were subscribed beforehand had not yet resubscribed when the next schema operation began. This was due to a stale list of subscribers being left behind in the schema distribution data; these subscribers were assumed by the coordinator to be participants in subsequent schema operations.

  We fix this issue by clearing the list of known subscribers whenever the `mysql.ndb_schema` table is dropped. (Bug #34843412)

- When requesting a new global checkpoint (GCP) from the data nodes, such as by the NDB Cluster handler in `mysqld` to speed up delivery of schema distribution events and responses, the request was sent 100 times. While the `DBDIH` block attempted to merge these duplicate requests into one, it was possible on occasion to trigger more than one immediate GCP. (Bug #34836471)

- When the `DBSPJ` block receives a query for execution, it sets up its own internal plan for how to do so. This plan is based on the query plan provided by the optimizer, with adaptions made to provide the most efficient execution of the query, both in terms of elapsed time and of total resources used.

  Query plans received by `DBSPJ` often contain star joins, in which several child tables depend on a common parent, as in the query shown here:

  ```
  SELECT STRAIGHT_JOIN * FROM t AS t1
  INNER JOIN t AS t2 ON t2.a = t1.k
  INNER JOIN t AS t3 ON t3.k = t1.k;
  ```

  In such cases `DBSPJ` could submit key-range lookups to `t2` and `t3` in parallel (but does not do so). An inner join also has the property that each inner joined row requires a match from the other tables in the same join nest, else the row is eliminated from the result set. Thus, by using the key-range lookups, we may retrieve rows from one such lookup which have no matches in the other, which effort is ultimately wasted. Instead, `DBSPJ` sets up a sequential plan for such a query.

  It was found that this worked as intended for queries having only inner joins, but if any of the tables are left-joined, we did not take complete advantage of the preceding inner joined tables before issuing the outer joined tables. Suppose the previous query is modified to include a left join, like this:

  ```
  SELECT STRAIGHT_JOIN * FROM t AS t1
  INNER JOIN t AS t2 ON t2.a = t1.k
  LEFT  JOIN t AS t3 ON t3.k = t1.k;
  ```

  Using the following query against the `ndbinfo.counters` table, it is possible to observe how many rows are returned for each query before and after query execution:

  ```
  SELECT counter_name, SUM(val)
  FROM ndbinfo.counters
  WHERE block_name="DBSPJ" AND counter_name = "SCAN_ROWS_RETURNED";
  ```

  It was thus determined that requests on `t2` and `t3` were submitted in parallel. Now in such cases, we wait for the inner join to complete before issuing the left join, so that unmatched rows from `t1` can be eliminated from the outer join on `t1` and `t3`. This results in less work to be performed by the data nodes, and reduces the volumne handled by the transporter as well. (Bug #34782276)

- SPJ handling of a sorted result was found to suffer a significant performance impact compared to the same result set when not sorted. Further investigation showed that most of the additional performance overhead for sorted results lay in the implementation for sorted result retrieval, which required an excessive number of `SCAN_NEXTREQ` round trips between the client and `DBSPJ` on the data nodes. (Bug #34768353)

- `DBSPJ` now implements the `firstMatch` optimization for semijoins and antijoins, such as those found in `EXISTS` and `NOT EXISTS` subqueries. (Bug #34768191)

- When the `DBSPJ` block sends `SCAN_FRAGREQ` and `SCAN_NEXTREQ` signals to the data nodes, it tries to determine the optimum number of fragments to scan in parallel without starting more parallel scans than needed to fill the available batch buffers, thus avoiding any need to send additional `SCAN_NEXTREQ` signals to complete the scan of each fragment.

  The `DBSPJ` block's statistics module calculates and samples the parallelism which was optimal for fragment scans just completed, for each completed `SCAN_FRAGREQ`, providing a mean and standard deviation of the sampled parallelism. This makes it possible to calculate a lower 95th percentile

of the parallelism (and batch size) which makes it possible to complete a SCAN_FRAGREQ without needing additional SCAN_NEXTREQ signals.

It was found that the parallelism statistics seemed unable to provide a stable parallelism estimate and that the standard deviation was unexpectedly high. This often led to the parallelism estimate being a negative number (always rounded up to 1).

The flaw in the statistics calculation was found to be an underlying assumption that each sampled SCAN_FRAGREQ contained the same number of key ranges to be scanned, which is not necessarily the case. Typically a full batch of rows for the first SCAN_FRAGREQ, and relatively few rows for the final SCAN_NEXTREQ returning the remaining rows; this resulted in wide variation in parallelism samples which made the statistics obtained from them unreliable.

We fix this by basing the statistics on the number of keys actually sent in the SCAN_FRAGREQ, and counting the rows returned from this request. Based on this it is possible to obtain record-per-key statistics to be calculated and sampled. This makes it possible to calculate the number of fragments which can be scanned, without overflowing the batch buffers. (Bug #34768106)

- It was possible in certain cases that both the NDB binary logging thread and metadata synchronization attempted to synchronize the ndb_apply_status table, which led to a race condition. We fix this by making sure that the ndb_apply_status table is monitored and created (or re-created) by the binary logging thread only. (Bug #34750992)

- While starting a schema operation, the client is responsible for detecting timeouts until the coordinator has received the first schema event; from that point, any schema operation timeout should be detected by the coordinator. A problem occurred while the client was checking the timeout; it mistakenly set the state indicating that timeout had occurred, which caused the coordinator to ignore the first schema event taking longer than approximately one second to receive (that is, to write the send event plus handle in the binary logging thread). This had the effect that, in these cases, the coordinator was not involved in the schema operation.

  We fix this by change the schema distribution timeout checking to be atomic, and to let it be performed by either the client or the coordinator. In addition, we remove the state variable used for keeping track of events received by the coordinator, and rely on the list of participants instead. (Bug #34741743)

- An SQL node did not start up correctly after restoring data with ndb_restore, such that, when it was otherwise ready to accept connections, the binary log injector thread never became ready. It was found that, when a mysqld was started after a data node initial restore from which new table IDs were generated, the utility table's (ndb_*) MySQL data dictionary definition might not match the NDB dictionary definition.

  The existing mysqld definition is dropped by name, thus removing the unique ndbcluster-*ID* key in the MySQL data dictionary but the new table ID could also already be occupied by another (stale) definition. The resulting mistmatch prevented setup of the binary log.

  To fix this problem we now explicitly drop any ndbcluster-*ID* definitions that might clash in such cases with the table being installed. (Bug #34733051)

- After receiving a SIGTERM signal, ndb_mgmd did not wait for all threads to shut down before exiting. (Bug #33522783)

  References: See also: Bug #32446105.

- When multiple operations are pending on a single row, it is not possible to commit an operation which is run concurrently with an operation which is pending abort. This could lead to data node shutdown during the commit operation in DBACC, which could manifest when a single transaction contained more than MaxDMLOperationsPerTransaction DML operations.

  In addition, a transaction containing insert operations is rolled back if a statement that uses a locking scan on the prepared insert fails due to too many DML operations. This could lead to an unplanned

data node shutdown during tuple deallocation due to a missing reference to the expected `DBLQH` deallocation operation.

We solve this issue by allowing commit of a scan operation in such cases, in order to release locks previously acquired during the transaction. We also add a new special case for this scenario, so that the deallocation is performed in a single phase, and `DBACC` tells `DBLQH` to deallocate immediately; in `DBLQH`, `execTUP_DEALLOCREQ()` is now able to handle this immediate deallocation request. (Bug #32491105)

References: See also: Bug #28893633, Bug #32997832.

- Cluster nodes sometimes reported `Failed to convert connection to transporter` warnings in logs, even when this was not really necessary. (Bug #14784707)
- When started with no connection string on the command line, `ndb_waiter` printed `Connecting to mgmsrv at (null)`. Now in such cases, it prints `Connecting to management server at nodeid=0,localhost:1186` if no other default host is specified.

  The `--help` option and other `ndb_waiter` program output was also improved. (Bug #12380163)
- `NdbSpin_Init()` calculated the wrong number of loops in `NdbSpin`, and contained logic errors. (Bug #108448, Bug #32497174, Bug #32594825)

  References: See also: Bug #31765660, Bug #32413458, Bug #102506, Bug #32478388.

Changes in MySQL NDB Cluster 8.0.32 (2023-01-17, General Availability)

MySQL NDB Cluster 8.0.32 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.32 (see Changes in MySQL 8.0.32 (2023-01-17, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Added the `--config-binary-file` option for `ndb_config`, which enables this program to read configuration information from the management server's binary configuration cache. This can be useful, for example, in determining whether or not the current version of the `config.ini` file has actually been read by the management server and applied to the cluster. See the description of the option in the MySQL NDB Cluster documentation, for more information and examples. (Bug #34773752)

Bugs Fixed

- **Packaging:** The man page for `ndbxfrm` was not present following installation. (Bug #34520046)
- **Solaris; NDB Client Programs:** `ndb_top` was not built for Solaris platforms. (Bug #34186837)
- **MySQL NDB ClusterJ:** ClusterJ could not be built on Ubuntu 22.10 with GCC 12.2. (Bug #34666985)

- In some contexts, a data node process may be sent `SIGCHLD` by other processes. Previously, the data node process bound a signal handler treating this signal as an error, which could cause the process to shut down unexpectedly when run in the foreground in a Kubernetes environment (and possibly under other conditions as well). This occurred despite the fact that a data node process never starts child processes itself, and thus there is no need to take action in such cases.

  To fix this, the handler has been modified to use `SIG_IGN`, which should result in cleanup of any child processes.

  Note

  `mysqld` and `ndb_mgmd` processes do not bind any handlers for `SIGCHLD`.

  (Bug #34826194)

- The running node from a node group scans each fragment (`CopyFrag`) and sends the rows to the starting peer in order to synchronize it. If a row from the fragment is locked exclusively by a user transaction, it blocks the scan from reading the fragment, causing the copyFrag to stall.

  If the starting node fails during the `CopyFrag` phase then normal node failure handling takes place. The cordinator node's transaction coordinator (TC) performs TC takeover of the user transactions from the TCs on the failed node. Since the scan that aids copying the fragment data over to the starting node is considered internal only, it is not a candidate for takeover, thus the takeover TC marks the `CopyFrag` scan as closed at the next opportunity, and waits until it is closed.

  The current issue arose when the `CopyFrag` scan was in the `waiting for row lock` state, and the closing of the marked scan was not performed. This led to TC takeover stalling while waiting for the close, causing unfinished node failure handling, and eventually a GCP stall potentially affecting redo logging, local checkpoints, and NDB Replication.

  We fix this by closing the marked `CopyFrag` scan whenever a node failure occurs while the `CopyFrag` is waiting for a row lock. (Bug #34823988)

  References: See also: Bug #35037327.

- In certain cases, invalid signal data was not handled correctly. (Bug #34787608)

- Sending of fragmented signals to virtual (`V_QUERY`) blocks is not supported, since the different signal fragments may end up in different block instances. When `DBTC` or `DBSPJ` sends a `LQHKEYREQ` or `SCAN_FRAGREQ` signal that may end up using `V_QUERY`, it checks whether the signal is fragmented and in that case changes the receiver to an instance of `DBLQH`. The function `SimulatedBlock::sendBatchedFragmentedSignal()` is intended to use the same check to decide whether to fragment a given signal, but did not, with the result that signals were fragmented which were not expected to be, sent using `V_QUERY`, and in that case likely to fail when received.

  We fix this problem by making the size check in `SimulatedBlock::sendFirstFragment()`, used by `sendBatchedFragmentedSignal()`, match the checks performed in `DBTC` and `DBSPJ`. (Bug #34776970)

- When the `DBSPJ` block submits `SCAN_FRAGREQ` requests to the local data managers, it usually scans only a subset of the fragments in parallel based on `recsPrKeys` statistics, if these are available, or just make a guess if no statistics are available.

  `SPJ` contains logic which may take advantage of the result collected from the first round of fragments scanned; parallelism statistics are collected after `SCAN_FRAGCONF` replies are received, and first-match elimination may eliminate keys needed to scan in subsequent rounds.

  Scanning local fragments is expected to have less overhead than scanning remote fragments, so it is preferable to err on the side of scan-parallelism for the local fragments. To take advantage of this, now two rounds are made over the fragments, the first one allowing `SCAN_FRAGREQ` signals to be

sent to local fragments only, the second allowing such singals to be sent to any fragment expecting it. (Bug #34768216)

References: See also: Bug #34768191.

- When pushing a join to the data nodes, the query request is distributed to the `SPJ` blocks of all data nodes having local fragments for the first table (the `SPJ` root) in the pushed query. Each `SPJ` block retrieves qualifying rows from the local fragments of this root table, then uses the retrieved rows to generate a request to its joined child tables. If no qualifying rows are retrieved from the local fragments of the root, `SPJ` has no further work to perform.

  This implies that for a pushed join in which the root returns few rows, there are likely to be idling `SPJ` workers not taking full advantage of the available parallelism. Now for such queries we do not include very small tables in the pushed join, so that, if the next table in the join plan is larger, we start with that one instead. (Bug #34723413)

- The safety check for a copying `ALTER TABLE` operation uses the sum of per-fragment commit count values to determine whether any writes have been committed to a given table over a period of time. Different replicas of the same fragment do not necessarily have the same commit count over time, since a fragment replica's commit count is reset during node restart.

  Read primary tables always route read requests to a table's primary fragment replicas. Read backup and fully replicated tables optimize reads by allowing `CommittedRead` operations to be routed to backup fragment replicas. This results in the set of commit counts read not always being stable for Read backup and fully replicated tables, which can cause false positive failures for the copying `ALTER TABLE` safety check.

  This is solved by performing the copying `ALTER TABLE` safety check using a locking scan. Locked reads are routed to the same set of primary (main) fragments every time, which causes these counts to be stable. (Bug #34654470)

- Following execution of `DROP NODEGROUP` in the management client, attempting to creating or altering an `NDB` table specifying an explicit number of partitions or using `MAX_ROWS` was rejected with `Got error 771 'Given NODEGROUP doesn't exist in this cluster' from NDB`. (Bug #34649576)

- `TYPE_NOTE_TRUNCATED` and `TYPE_NOTE_TIME_TRUNCATED` were treated as errors instead of being ignored, as was the case prior to NDB 8.0.27. This stopped building of interpreted code for pushed conditions, with the condition being returned to the server.

  We fix this by reverting the handling of these status types to ignoring them, as was done previously. (Bug #34644930)

- When reorganizing a table with `ALTER TABLE ... REORGANIZE PARTITION` following addition of new data nodes to the cluster, fragments were not redistributed properly when the `ClassicFragmentation` configuration parameter was set to `OFF`. (Bug #34640773)

- Fixed an uninitialized padding variable in `src/common/util/ndb_zlib.cpp`. (Bug #34639073)

- When the `NDB_STORED_USER` privilege was granted to a user with an empty password, the user's password on each of the other SQL nodes was expired. (Bug #34626727)

- In a cluster with multiple management nodes, when one management node connected and later disconnected, any remaining management nodes were not aware of this node and were eventually forced to shut down when stopped nodes reconnected; this happened whenever the cluster still had live data nodes.

  On investigation it was found that node disconnection handling was done in the `NF_COMPLETEREP` path in `ConfigManager` but the expected `NF_COMPLETEREP` signal never actually arrived. We solve this by handling disconnecting management nodes when the `NODE_FAILREP` signal arrives, rather than waiting for `NF_COMPLETEREP`. (Bug #34582919)

- The `--diff-default` option and related options for `ndb_config` did not produce any usable output. (Bug #34549189)

  References: This issue is a regression of: Bug #32233543.

- Encrypted backups created on a system using one endian could not be restored on systems with the other endian; for example, encrypted backups taken on an x86 system could not be restored on a SPARC system, nor the reverse. (Bug #34446917)

- A query using a pushed join with an `IN` subquery did not return the expected result with `ndb_join_pushdown=ON` and the `BatchSize` SQL node parameter set to a very small value such as `1`. (Bug #34231718)

- When defining a binary log transaction, the transaction is kept in an in-memory binary log cache before it is flushed to the binary log file. If a binary log transaction exceeds the size of the cache, it is written to a temporary file which is set up early in the initialization of the binary log thread. This write introduces extra disk I/O in the binary log injector path. The number of disk writes performed globally by the binary log injector can be found by checking the value of the `Binlog_cache_disk_use` system status variable, but otherwise, the `NDB` handler's binary log injector thread had no way to observe this.

  Since `Binlog_cache_disk_use` is accessible by the binary log injector, it can be checked both before and after the transaction is committed to see whether there were any changes to its value. If any cache spills have taken place, this is reflected by the difference of the two values, and the binary log injector thread can report it. (Bug #33960014)

- When closing a file using compressed or encrypted format after reading the entire file, verify its checksum. (Bug #32550145)

- When reorganizing a table with `ALTER TABLE ... REORGANIZE PARTITION` following addition of new data nodes to the cluster, unique hash indexes were not redistributed properly. (Bug #30049013)

- For a partial local checkpoint, each fragment LCP must be to be able to determine the precise state of the fragment at the start of the LCP and the precise difference in the fragment between the start of the current LCP and the start of the previous one. This is tracked using row header information and page header information; in cases where physical pages are removed this is also tracked in logical page map information.

  A page included in the current LCP, before the LCP scan reaches it, is released due to the commit or rollback of some operation on the fragment, also releasing the last used storage on the page.

  Since the released page could not be found by the scan, the release itself set the `LCP_SCANNED_BIT` of the page map entry it was mapped into, in order to indicate that the page was already handled from the point of view of the current LCP, causing subsequent allocation and release of the pages mapped to the entry during the LCP to be ignored. The state of the entry at the start of the LCP was also set as allocated in the page map entry.

  These settings are cleared only when the next LCP is prepared. Any page release associated with the page map entry before the clearance would violate the requirement that the bit is not set; we resolve this issue by removing the (incorrect) requirement. (Bug #23539857)

- A data node could hit an overly strict assertion when the thread liveness watchdog triggered while the node was already shutting down. We fix the issue by relaxing this assertion in such cases. (Bug #22159697)

- Removed a leak of long message buffer memory that occurred each time an index was scanned for updating index statistics. (Bug #108043, Bug #34568135)

- `Backup::get_total_memory()`, used to calculate proposed disk write speeds for checkpoints, wrongly considered `DataMemory` that may not have been used in the calculation of memory used by LDMs.

  We fix this by obtaining the total `DataMemory` used by the LDM threads instead. as reported by `DBTUP`. (Bug #106907, Bug #34035805)

- Fixed an uninitialized variable in `Suma.cpp`. (Bug #106081, Bug #33764143)

Changes in MySQL NDB Cluster 8.0.31 (2022-10-11, General Availability)

MySQL NDB Cluster 8.0.31 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.31 (see Changes in MySQL 8.0.31 (2022-10-11, General Availability)).

- Transparent Data Encryption (TDE)
- RPM Notes
- Functionality Added or Changed
- Bugs Fixed

Transparent Data Encryption (TDE)

- This release implements Transparent Data Encryption (TDE), which provides protection by encryption of `NDB` data at rest. This includes all `NDB` table data and log files which are persisted to disk, and is intended to protect against recovering data subsequent to unauthorized access to NDB Cluster data files such as tablespace files or logs.

  To enforce encryption on files storing `NDB` table data, set `EncryptedFileSystem` to `1`, which causes all data to be encrypted and decrypted as necessary, as it is written to and read from these files. These include LCP data files, redo log files, tablespace files, and undo log files.

  When using file system encryption with `NDB`, you must also perform the following tasks:

  - Provide a password to each data node when starting or restarting it, using either one of the data node options `--filesystem-password` or `--filesystem-password-from-stdin`. This password uses the same format and is subject to the same constraints as the password used for an encrypted `NDB` backup (see the description of the `ndb_restore --backup-password` option).

    You can provide the encryption password on the command line, or in a `my.cnf` file. See NDB File System Encryption Setup and Usage, for more information and examples.

  Only tables using the `NDB` storage engine are subject to encryption by this feature; see NDB File System Encryption Limitations. Other tables, such as those used for `NDB` schema distribution, replication, and binary logging, typically use `InnoDB`; see InnoDB Data-at-Rest Encryption. For information about encryption of binary log files, see Encrypting Binary Log Files and Relay Log Files.

Files generated or used by NDB processes, such as operating system logs, crash logs, and core dumps, are not encrypted. Files used by NDB but not containing any user table data are also not encrypted; these include LCP control files, schema files, and system files (see NDB Cluster Data Node File System). The management server configuration cache is also not encrypted.

In addition, NDB 8.0.31 adds a new utility ndb_secretsfile_reader for extracting key information from encrypted files.

This enhancement builds on work done in NDB 8.0.22 to implement encrypted NDB backups. For more information, see the description of the RequireEncryptedBackup configuration parameter, as well as Using The NDB Cluster Management Client to Create a Backup.

Note

Upgrading an encrypted filesystem to NDB 8.0.31 or later from a previous release requires a rolling initial restart of the data nodes, due to improvements in key handling.

(Bug #34417282, WL #14687, WL #15051, WL #15204)

RPM Notes

- **ndbinfo Information Database:** Upgrades of SQL nodes from NDB 7.5 or NDB 7.6 to NDB 8.0 using RPM files did not enable the ndbinfo plugin properly. This was due to the fact that, since the ndbcluster plugin is disabled during an upgrade of mysqld, so is the ndbinfo plugin; this led to .frm files associated with ndbinfo tables being left behind following the upgrade.

  Now in such cases, any ndbinfo table .frm files from the earlier release are removed, and the plugin enabled. (Bug #34432446)

Functionality Added or Changed

- **Important Change; NDB Client Programs:** A number of NDB program options were never implemented and have now been removed. The options and the programs from which they have been dropped are listed here:

  - --ndb-optimized-node-selection: ndbd, ndbmtd, ndb_mgm, ndb_delete_all, ndb_desc, ndb_drop_index, ndb_drop_table, ndb_show_tables, ndb_blob_tool, ndb_config, ndb_index_stat, ndb_move_data, ndbinfo_select_all, ndb_select_count
  - --character-sets-dir: ndb_mgm, ndb_mgmd, ndb_config, ndb_delete_all, ndb_desc, ndb_drop_index, ndb_drop_table, ndb_show_tables, ndb_blob_tool, ndb_config, ndb_index_stat, ndb_move_data, ndbinfo_select_all, ndb_select_count, ndb_waiter
  - --core-file: ndb_mgm, ndb_mgmd, ndb_config, ndb_delete_all, ndb_desc, ndb_drop_index, ndb_drop_table, ndb_show_tables, ndb_blob_tool, ndb_config, ndb_index_stat, ndb_move_data, ndbinfo_select_all, ndb_select_count, ndb_waiter
  - --connect-retries and --connect-retry-delay: ndb_mgmd
  - --ndb-nodeid: ndb_config

  See the descriptions of the indicated program and program options in NDB Cluster Programs, for more information. (Bug #34059253)

- **Important Change:** The ndbcluster plugin is now included in all MySQL server builds, with the exception of builds for 32-bit platforms. As part of this work, we address a number of issues with

`cmake` options for NDB Cluster, making the plugin option for `NDBCLUSTER` behave as other plugin options, and adding a new option `WITH_NDB` to control the build of NDB for MySQL Cluster.

This release makes the following changes in `cmake` options relating to MySQL Cluster:

- Adds the `WITH_NDB` option (default `OFF`). Enabling this option causes the MySQL Cluster binaries to be built.
- Deprecates the `WITH_NDBCLUSTER` option; use `WITH_NDB` instead.
- Removes the `WITH_PLUGIN_NDBCLUSTER` option. Use `WITH_NDB`, instead, to build MySQL Cluster.
- Changes the `WITH_NDBCLUSTER_STORAGE_ENGINE` option so that it now controls (only) whether the `ndbcluster` plugin itself is built. This option is now automatically set to `ON` when `WITH_NDB` is enabled for the build, so it should no longer be necessary to set it when compiling MySQL with NDB Cluster support.

For more information, see CMake Options for Compiling NDB Cluster. (WL #14788, WL #15157)

- Added the `--detailed-info` option for `ndbxfrm`. This is similar to the `--info` option, but in addition prints out the file's header and trailer. (Bug #34380739)
- This release makes it possible to enable and disable binary logging with compressed transactions using `ZSTD` compression for `NDB` tables in a `mysql` or other client session while the MySQL server is running. To enable the feature, set the `ndb_log_transaction_compression` system variable introduced in this release to `ON`. The level of compression used can be controlled using the `ndb_log_transaction_compression_level_zstd` system variable, which is also added in this release; the default compression level is `3`.

  Note

  Although changing the values of the `binlog_transaction_compression` and `binlog_transaction_compression_level_zstd` system variables from a client session has no effect on binary logging of `NDB` tables, setting `--binlog-transaction-compression=ON` on the command line or in a `my.cnf` file causes `ndb_log_transaction_compression` to be enabled, regardless of any setting for `--ndb-log-transaction-compression`. In this case, to disable binary log transaction compression for (only) `NDB` tables, set `ndb_log_transaction_compression=OFF` in a MySQL client session following startup of `mysqld`.

  For more information, see Binary Log Transaction Compression. (Bug #32704705, Bug #32927582, WL #15138, WL #15139)

Bugs Fixed

- When pushing a condition as part of a pushed join, it is a requirement that all *`table.column`* references are to one of the following:
  - The table to which the condition itself is pushed
  - A table which is an ancestor of the root of the pushed join
  - A table which is an ancestor of the table in the pushed query tree

  In the last case, when finding possible ancestors, we did not fully identify all candidates for such tables, in either or both of these two ways:

  1. Any tables being required ancestors due to nest-level dependencies were not added as ancestors

2. Tables having all possible ancestors as either required ancestors or key parents are known to be directly joined with our ancestor, and to provide these as ancestors themselves; thus, such tables should be made available as ancestors as well.

This patch implements both cases 1 and 2. In the second case, we take a conservative approach and add only those tables having a `single row lookup` access type, but not those using index scans. (Bug #34508948)

- Execution of and `EXPLAIN` for some large join queries with ndb_join_pushdown enabled (the default) were rejected with NDB error `QRY_NEST_NOT_SUPPORTED` `FirstInner/Upper has to be an ancestor or a sibling`. (Bug #34486874)

- When the `NDB` join pushdown handler finds a table which cannot be pushed down it tries to produce an explanatory message communicating the reason for the rejection, which includes the names of the tables involved. In some cases the optimizer had already optimized away the table which meant that it could no longer be accessed by the `NDB` handler, resulting in failure of the query.

  We fix this by introducing a check for such cases and printing a more generic message which does not include the table name if no table is found. (Bug #34482783)

- The `EncryptedFilesystem` parameter was not defined with `CI_RESTART_INITIAL`, and so was not shown in the output of `ndb_config` as requiring `--initial`, even though the parameter does in fact require an initial restart to take effect. (Bug #34456384)

- When finding tables possible to push down in a pushed join, the pushability of a table may depend on whether later tables are pushed as well. In such cases we take an optimistic approach and assume that later tables are also pushed. If this assumption fails, we might need to “unpush” a table and any other tables depending on it. Such a cascading “unpush” may be due to either or both of the following conditions:

  - A key reference referred to a column from a table which later turned out to not be pushable.

  - A pushed condition referred to a column from a table which later turn out to not be pushable.

  We previously handled the first case, but handling of the second was omitted from work done in NDB 8.0.27 to enable pushing of conditions referring to columns from other tables that were part of the same pushed join. (Bug #34379950)

- `NdbScanOperation` errors are returned asynchronously to the client, possibly while the client is engaged in other processing. A successful call to `NdbTransaction::execute()` guarantees only that the scan request has been assembled and sent to the transaction coordinator without any errors; it does not wait for any sort of `CONF` or `REF` signal to be returned from the data nodes. In this particular case, the expected `TAB_SCANREF` signal was returned asynchronously into the client space, possibly while the client was still performing other operations.

  We make this behavior more deterministic by not setting the `NdbTransaction` error code when a `TAB_SCANREF` error is received. (Bug #34348706)

- When attempting to update a `VARCHAR` column that was part of an `NDB` table's primary key, the length of the value read from the database supplied to the `cmp_attr()` method was reportedly incorrectly. In addition to fixing this issue, we also remove an incorrect length check which required the binary byte length of the arguments to this method to be the same, which is not true of attributes being compared as characters, whose comparison semantics are defined by their character sets and collations. (Bug #34312769)

- When compiling NDB Cluster on OEL7 and OEL8 using `-Og` for debug builds, `gcc` raised a null pointer subtraction error. (Bug #34199675, Bug #34199732)

  References: See also: Bug #33855533.

- `ndb_blob_tool` did not perform proper handling of errors raised while reading data. (Bug #34194708)

- As part of setting up the signal execution strategy, we calculate a safe quota for the maximum numbers signals to execute from each job buffer. As each executed signal is assumed to generate up to four outward bound signals, we might need to limit the signal quota so that we do not overfill the out buffers. Effectively, in each round of signal execution we cannot execute more signals than 1/4 of the signals that can fit in the out buffers.

  This calculation did not take into account work done in NDB 8.0.23 introducing the possibility of having multiple writers, all using the same available free space in the same job buffer. Thus the signal quota needed to be further divided among the workers writing to the same buffers.

  Now the computation of the maximum numbers signals to execute takes into account the resulting possibly greater number of writers to each queue. (Bug #34065930)

- When the `NDB` scheduler detects that job buffers are full, and starts to allocate from reserved buffers, it is expected to yield the CPU while waiting for the consumer. Just before yielding, it performs a final check for this condition, before sleeping. Problems arose when this check indicated that the job buffers were not full, so that the scheduler was allowed to continue executing signals, even though the limit on how many signals it was permitted to execute was still `0`. This led to a round of executing no signals, followed by another yield check, and so on, keeping the CPU occupied for no reason while waiting for something to be consumed by the receiver threads.

  The root cause of the problem was that different metrics were employed for calculating the limit on signals to execute (which triggered the yield check when this limit was `0`), and for the yield callback which subsequently checked whether the job buffers were actually full.

  Prior to the implementation of scalable job buffers in MySQL NDB Cluster 8.0.23, `NDB` waited for more job buffer up to 10 times; this was inadvertently changed so that it gave up after waiting one time only, despite log messages indicating that `NDB` had slept ten times. As part of this fix, we revert that change, so that, as before, we wait up to ten times for more job buffer before giving up. As an additional part of this work, we also remove extra (and unneeded) code previously added to detect spin waits. (Bug #34038016)

  References: See also: Bug #33869715, Bug #34025532.

- Job buffers act as the communication links between data node internal block threads. When the data structures for these were initialized, a 32K page was allocated to each such link, even if these threads never (by design) communicate with each other. This wasted memory resources, and had a small performance impact since the job buffer pages were checked frequently for available signals, so that us was necessary to load the unused job buffer pages into the translation lookaside buffer and L1, L2, and L3 caches.

  Now, instead, we set up an empty job buffer as a sentinel to which all the communication links refer initially. Actual (used) job buffer pages are now allocated only when we actually write signals into them, in the same way that new memory pages are allocated when a page gets full. (Bug #34032102)

- A data node could be forced to shut down due to a full job buffer, even when the local buffer was still available. (Bug #34028364)

- Made checks of pending signals by the job scheduler more consistent and reliable. (Bug #34025532)

  References: See also: Bug #33869715, Bug #34038016.

- The combination of batching with multiple in-flight operations per key, use of `IgnoreError`, and transient errors occurring on non-primary replicas led in some cases to inconsistencies within `DBTUP` resulting in replica misalignment and other issues. We now prevent this from happening by detecting when operations are failing on non-primary replicas, and forcing `AbortOnError` handling (rollback) in such cases for the containing transaction. (Bug #34013385)

- Handling by `ndb_restore` of temporary errors raised by DDL operations has been improved and made consistent. In all such cases, `ndb_restore` now retries the operation up to `MAX_RETRIES` (11) times before giving up. (Bug #33982499)

- Removed the causes of many warnings raised when compiling NDB Cluster. (Bug #33797357, Bug #33881953)

- When the rate of changes was high, event subscribers were slow to acknowledge receipt, or both, it was possible for the `SUMA` block to run out of space for buffering events. (Bug #30467140)

- `ALTER TABLE ... COMMENT="NDB_TABLE=READ_BACKUP=1"` or `ALTER TABLE..COMMENT="NDB_TABLE=READ_BACKUP=0"` performs a non-copying (online) `ALTER` operation on a table to add or remove its `READ_BACKUP` property (see NDB_TABLE Options), which increments the index version of all indexes on the table. Existing statistics, stored using the previous index version, were orphaned and never deleted; this led to wasted memory and inefficient searches when collecting index statistics.

  We address these issues by cleaning up the index samples; we delete any samples whose sample version is greater than or less than the current sample version. In addition, when no existing statistics are found by index ID and version, and when indexes are dropped. In this last case, we relax the bounds for the delete operation and remove all entries corresponding to the index ID in question, as opposed to both index ID and index version.

  This fix cleans up the sample table which stores the bulk of index statistics data. The head table, which consists of index metadata rather than actual statistics, still contains orphaned rows, but since these occupy an insignificant amount of memory, they do not adversely affect statistics search efficiency, and stale entries are cleaned up when index IDs and versions are reused.

  See also NDB API Statistics Counters and Variables. (Bug #29611297)

Changes in MySQL NDB Cluster 8.0.30 (2022-07-26, General Availability)

MySQL NDB Cluster 8.0.30 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.30 (see Changes in MySQL 8.0.30 (2022-07-26, General Availability)).

- Functionality Added or Changed

- Bugs Fixed

Functionality Added or Changed

- **Important Change; NDB Replication:** The `replica_allow_batching` system variable affects how efficiently the replica applies epoch transactions. When this is set to `OFF`, by default, every discrete replication event in the binary log is applied and executed separately, which generally leads to poor performance.

  Beginning with this release, the default value for `replica_allow_batching` is changed from `OFF` to `ON`.

- **NDB Replication:** NDB Cluster Replication supports conflict detection and resolution for use in circular replication setups to determine whether or not to apply a given operation when it uses the same primary key as another operation. Previously, it was possible to resolve primary key conflicts for update and delete operations, but for write operations with the same primary key, the only handling performed was to reject any write operation having the same primary key as an existing one, and to apply only if no write operation existed that had the same primary key exists.

  This release introduces two new conflict resolution functions `NDB$MAX_INS()` and `NDB$MAX_DEL_WIN_INS()`. Each of these functions provides handling of primary key conflicts between insert (write) operations following the steps shown here:

  1. If there is no conflicting write, apply this one (this is the same behavior as `NDB$MAX()` and `NDB$MAX_DELETE_WIN()`).
  2. In the event of a conflict, apply “greatest timestamp wins” conflict resolution, as follows:
     a. If the timestamp for the incoming write is greater than that of the conflicting write, apply the incoming write operation.
     b. If the timestamp for the incoming write is *not* greater, reject the incoming write operation.

  `NDB$MAX_INS()` handles conflicting update and delete operations in the same way as `NDB$MAX()`, and `NDB$MAX_DEL_WIN_INS()` does so in the same way as `NDB$MAX_DELETE_WIN()`.

  This enhancement provides support for configuring conflict detection when handling conflicting replicated write operations, so that a replicated `INSERT` with a higher timestamp column value is applied idempotently, and a replicated `INSERT` with a lower timestamp column value is rejected.

  As with the other conflict resolution functions, rejected operations can optionally be logged in an exceptions table, and rejected operations increment a counter; for “greater timestamp wins” handling, this is the status variable `Ndb_conflict_fn_max_ins`, and for the “same timestamp wins” strategy, the counter incremented is `Ndb_conflict_fn_max_del_win_ins`.

  For more information, see Conflict Resolution Functions. (Bug #33398980, WL #14942)

- **NDB Replication:** Previously, the size of batches used when writing to a replica NDB Cluster was controlled by the `--ndb-batch-size`, and the batch size used for writing blob data to the replica was determined by `ndb-blob-write-batch-bytes`. This scheme led to issues due to the fact that the replica used the global values of these variables; this meant that changing either or both of them on the replica also affected the values used by all other sessions. Another shortcoming of this arrangement was that it was not possible to set different default for these options exclusive to the replica applier, which should preferably have a higher default value than other sessions.

  This release solves these problems by adding two new system variables which are specific to the replica applier. `ndb_replica_batch_size` now controls the batch size used for the replica applier, and `ndb_replica_blob_write_batch_bytes` variable now determines the blob write batch size used to perform batch writes of blobs on the replica.

  This change should improve the behavior of MySQL NDB Cluster Replication using default settings, and lets the user fine tune NDB replication without affecting user threads performing other tasks, such as processing SQL queries.

  For more information, see the descriptions of the new system variables. See also Preparing the NDB Cluster for Replication. (WL #15070, WL #15071)

  References: See also: Bug #21040523.

- **NDB Cluster APIs:** Removed a number of potential memory leaks by using `std::uniqe_ptr` for managing any `Event` returned by `Dictionary::getEvent()`.

  As part of this fix, we add a `releaseEvent()` method to `Dictionary` to clean up events created with `getEvent()` after they are no longer needed. (Bug #33855045)

- **NDB Cluster APIs:** The `Node.JS` package included with NDB Cluster has been updated to version 16.5.0. (Bug #33770627)

- Empty lines in CSV files are now accepted as valid input by `ndb_import`. (Previously, empty lines in such files were always rejected.) Now, if an empty value can be used as the value for a single imported column, `ndb_import` uses it in the same manner as `LOAD DATA`. (Bug #34119833)

- `NDB` stores blob column values differently from other types; by default, only the first 256 bytes of the value are stored in the table ("inline"), with any remainder kept in a separate blob parts table. This is true for columns of MySQL type `BLOB`, `MEDIUMBLOB`, `LONGBLOB`, `TEXT`, `MEDIUMTEXT`, and `LONGTEXT`. (`TINYBLOB` and `TINYTEXT` are exceptions, since they are always inline-only.) `NDB` handles `JSON` column values in a similar fashion, the only difference being that, for a `JSON` column, the first 4000 bytes of the value are stored inline.

  Previously, it was possible to control the inline size for blob columns of `NDB` tables only by using the NDB API (`Column::setInlineSize()` method). This now can be done in the `mysql` client (or other application supplying an SQL interface) using a column comment which consists of an `NDB_COLUMN` string containing a `BLOB_INLINE_SIZE` specification, as part of a `CREATE TABLE` statement like this one:

  ```
  CREATE TABLE t (
      a BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
      b BLOB COMMENT 'NDB_COLUMN=BLOB_INLINE_SIZE=3000'
  ) ENGINE NDBCLUSTER;
  ```

  In table `t` created by the statement just shown, column `b` (emphasized text in the preceding example) is a `BLOB` column whose first 3000 bytes are stored in `t` itself, rather than just the first 256 bytes. This means that, if no value stored in `b` exceeds 3000 bytes in length, no extra work is required to read or write any excess data from the `NDB` blob parts table when storing or retrieving the column value. This can improve performance significantly when performing many operations on blob columns.

  You can see the effects of this option by querying the `ndbinfo.blobs` table, or examining the output of `ndb_desc`.

  The maximum supported value for `BLOB_INLINE_SIZE` is 29980. Setting it to any value less than 1 causes the default inline size to be used for the column.

  You can also alter a column as part of a copying `ALTER TABLE`; `ALGORITHM=INPLACE` is not supported for such operations.

  `BLOB_INLINE_SIZE` can be used alone, or together with `MAX_BLOB_PART_SIZE` in the same `NDB_COMMENT` string. Unlike the case with `MAX_BLOB_PART_SIZE`, setting `BLOB_INLINE_SIZE` is supported for `JSON` columns of `NDB` tables.

  For more information, see NDB_COLUMN Options, as well as String Type Storage Requirements. (Bug #33755627, WL #15044)

- A new `--missing-ai-column` option is added to `ndb_import`. This enables `ndb_import` to accept a CSV file from which the data for an `AUTO_INCREMENT` column is missing and to supply these values itself, much as `LOAD DATA` does. This can be done with one or more tables for which the CSV representation contains no values for such a column.

  This option works only when the CSV file contains no nonempty values for the `AUTO_INCREMENT` column to be imported. (Bug #102730, Bug #32553029)

- This release adds Performance Schema instrumentation for transaction batch memory used by `NDBCLUSTER`, making it possible to monitor memory used by transactions. For more information, see Transaction Memory Usage. (WL #15073)

Bugs Fixed

- **Important Change:** When using the `ThreadConfig` multithreaded data node parameter to specify the threads to be created on the data nodes, the receive thread (`recv`) in some cases was placed in the same worker thread as block threads such as `DBLQH(0)` and `DBTC(0)`. This represented a regression from NDB 8.0.22 and earlier, where the receive thread is colocated only with `THRMAN` and `TRPMAN`, as expected in such cases.

  Now, when setting the value of `ThreadConfig`, you must include `main`, `rep`, `recv`, and `ldm` explicitly; to avoid using one or more of the `main`, `rep`, or `ldm` thread types, you must set `count=0` explicitly for each applicable thread type.

  In addition, a minimum value of `1` is now enforced for the `recv` count; setting the replication thread (`rep`) count to `1` also requires setting `count=1` for the main thread.

  These changes can have serious implications for upgrades from previous NDB Cluster releases. For more information, see Upgrading and Downgrading NDB Cluster, as well as the description of the `ThreadConfig` parameter, in the MySQL Manual. (Bug #33869715)

  References: See also: Bug #34038016, Bug #34025532.

- **macOS:** `ndb_import` could not be compiled on MacOS/ARM because the `ndbgeneral` library was not explicitly included in `LINK_LIBRARIES`. (Bug #33931512)

- **NDB Disk Data:** The `LGMAN` kernel block did not initialize its local encrypted filesystem state, and did not check `EncryptedFileSystem` for undo log files, so that their encryption status was never actually set.

  This meant that, for release builds, it was possible for the undo log files to be encrypted on some systems, even though they should not have been; in debug builds, undo log files were always encrypted. This could lead to problems when using Disk Data tables and upgrading to or from NDB 8.0.29. (A workaround is to perform initial restarts of the data nodes when doing so.)

  This issue could also cause unexpected CPU load for I/O threads when there were a great many Disk Data updates to write to the undo log, or at data node startup while reading the undo log.

  > **Note**
  >
  > The `EncryptedFileSystem` parameter, introduced in NDB 8.0.29, is considered experimental and is not supported in production.

  (Bug #34185524)

- **NDB Replication:** Updating a row of an `NDB` table having only the hidden primary key (but no explicit PK) on the source could lead to a stoppage of the SQL thread on the replica. (Bug #33974581)

- **NDB Cluster APIs:** The internal function `NdbThread_SetThreadPrio()` sets the thread priority (`thread_prio`) for a given thread type when applying the setting of the `ThreadConfig` configuration parameter. It was possible for this function in some cases to return an error when it had actually succeeded, which could have a an unfavorable impact on the performance of some NDB API applications. (Bug #34038630)

- **NDB Cluster APIs:** The following `NdbInterpretedCode` methods did not function correctly when a nonzero value was employed for the *`label`* argument:

  - `branch_col_and_mask_eq_mask()`

- `branch_col_and_mask_eq_zero()`
- `branch_col_and_mask_ne_mask()`
- `branch_col_and_mask_ne_zero()`

(Bug #33888962)

- **MySQL NDB ClusterJ:** ClusterJ support for systems with ARM-based Apple silicon is now enabled by default. (Bug #34148474)
- Compilation of NDB Cluster on Debian 11 and Ubuntu 22.04 halted during the link time optimization phase due to source code warnings being treated as errors. (Bug #34252425)
- `NDB` does not support in-place changes of default values for columns; such changes can be made only by using a copying `ALTER TABLE`. Changing of the default value in such cases was already detected, but the additional or removal of default value was not.

  We fix this issue by detecting when default value is added or removed during `ALTER TABLE`, and refusing to perform the operation in place. (Bug #34224193)
- After creating a user on SQL node A and granting it the `NDB_STORED_USER` privilege, dropping this user from SQL node B led to inconsistent results. In some cases, the drop was not distributed, so that after the drop the user still existed on SQL node A.

  The cause of this issue is that `NDB` maintains a cache of all local users with `NDB_STORED_USER`, but when a user was created on SQL node B, this cache was not updated. Later, when executing `DROP USER`, this led SQL node B to determine that the drop did not have to be distributed. We fix this by ensuring that this cache is updated whenever a new distributed user is created. (Bug #34183149)
- When the internal `ndbd_exit()` function was invoked on a data node, information and error messages sent to the event logger just prior to the `ndbd_exit()` call were not printed in the log as expected. (Bug #34148712)
- NDB Cluster did not compile correctly on Ubuntu 22.04 due to changes in OpenSSL 3.0. (Bug #34109171)
- NDB Cluster would not compile correctly using GCC 8.4 due to a change in Bison fallthrough handling. (Bug #34098818)
- Compiling the `ndbcluster` plugin or the `libndbclient` library required a number of files kept under directories specific to data nodes (`src/kernel`) and management servers (`src/mgmsrv`). These have now been moved to more suitable locations. Files moved that may be of interest are listed here:

  - `ndbd_exit_codes.cpp` is moved to `storage/ndb/src/mgmapi`
  - `ConfigInfo.cpp` is moved to `storage/ndb/src/common/mgmcommon`
  - `mt_thr_config.cpp` is moved to `storage/ndb/src/common`
  - `NdbinfoTables.cpp` is moved to `storage/ndb/src/common/debugger`

  (Bug #34045289)
- When an error occurred during the begin schema transaction phase, an attempt to update the index statistics automatically was made without releasing the transaction handle, leading to a leak. (Bug #34007422)

  References: See also: Bug #34992370.
- Path lengths were not always calculated correctly by the data nodes. (Bug #33993607)

- When `ndb_restore` performed an NDB API operation with any concurrent NDB API events taking place, contention could occur in the event of limited resources in `DBUTIL`. This led to temporary errors in `NDB`. In such cases, `ndb_restore` now attempts to retry the NDB API operation which failed. (Bug #33984717)

  References: See also: Bug #33982499.

- Removed a duplicate check of a table pointer found in the internal method `Dbtc::execSCAN_TABREQ()`. (Bug #33945967)

- The internal function `NdbReceiver::unpackRecAttr()`, which unpacks attribute values from a buffer from a `GSN_TRANSID_AI` signal, did not check to ensure that attribute sizes fit within the buffer. This could corrupt the buffer which could in turn lead to reading beyond the buffer and copying beyond destination buffers. (Bug #33941167)

- Improved formatting of log messages such that the format string verification employed by some compilers is no longer bypassed. (Bug #33930738)

- Some `NDB` internal signals were not always checked properly. (Bug #33896428)

- Fixed a number of issues in the source that raised `-Wunused-parameter` warnings when compiling NDB Cluster with GCC 11.2. (Bug #33881953)

- When an SQL node was not yet connected to `NDBCLUSTER`, an excessive number of warnings were written to the MySQL error log when the SQL node could not discover an `NDB` table. (Bug #33875273)

- The NDB API statistics variables `Ndb_api_wait_nanos_count`, `Ndb_api_wait_nanos_count_replica`, and `Ndb_api_wait_nanos_count_session` are used for determining CPU times and wait times for applications. These counters are intended to show the time spent waiting for responses from data nodes, but they were not entirely accurate because time spent waiting for key requests was not included.

  For more information, see NDB API Statistics Counters and Variables. (Bug #33840016)

  References: See also: Bug #33850590.

- It was possible in some cases for a duplicate `engine`-`se_private_id` entry to be installed in the MySQL data dictionary for an `NDB` table, even when the previous table definition should have been dropped.

  When data nodes drop out of the cluster and need to rejoin, each SQL node starts to synchronize the schema definitions in its own data dictionary. The `se_private_id` for an `NDB` table installed in the data dictionary is the same as its `NDB` table ID. It is common for tables to be updated with different IDs, such as when executing an `ALTER TABLE`, `DROP TABLE`, or `CREATE TABLE` statement. The previous table definition, obtained by referencing the table in *`schema.table`* format, is usually sufficient for a drop and thus for the new table to be installed with a new ID, since it is assumed that no other installed table definition uses that ID. An exception to this could occur during synchronization, if a data node shutdown allowed the previous table definition of a table having the same ID other than the one to be installed to remain, then the old definition was not dropped.

  To correct this issue, we now check whether the ID of the table to be installed in the data dictionary differs from that of the previous one, in which case we also check whether an old table definition already exists with that ID, and, if it does, we drop the old table before continuing. (Bug #33824058)

- After receiving a `COPY_FRAGREQ` signal, `DBLQH` sometimes places the signal in a queue by copying the signal object into a stored object. Problems could arise when this signal object was used to send another signal before the incoming `COPY_FRAGREQ` was stored; this led to saving a corrupt signal that, when sent, prevented a system restart from completing. We fix this by using a static copy of the signal for storage and retrieval, rather than the original signal object. (Bug #33581144)

- When the `mysqld` binary supplied with NDB Cluster was run without `NDB` support enabled, the `ndbinfo` and `ndb_transid_mysql_connection_map` plugins were still enabled, and for example, still shown with status `ACTIVE` in the output of `SHOW PLUGINS`. (Bug #33473346)
- Attempting to seize a redo log page could in theory fail due to a wrong bound error. (Bug #32959887)
- When a data node was started using the `--foreground` option, and with a node ID not configured to connect from a valid host, the data node underwent a forced shutdown instead of reporting an error. (Bug #106962, Bug #34052740)
- `NDB` tables were skipped in the MySQL Server upgrade phase and were instead migrated by the `ndbcluster` plugin at a later stage. As a result, triggers associated with `NDB` tables were not created during upgrades from 5.7 based versions.

  This occurred because it is not possible to create such triggers when the `NDB` tables are migrated by the `ndbcluster` plugin, since metadata about the triggers is lost in the upgrade finalization phase of the MySQL Server upgrade in which all `.TRG` files are deleted.

  To fix this issue, we make the following changes:

  - Migration of `NDB` tables with triggers is no longer deferred during the Server upgrade phase.
  - `NDB` tables with triggers are no longer removed from the data dictionary during setup even when initial system starts are detected.

  (Bug #106883, Bug #34058984)
- When initializing a file, `NDBFS` enabled autosync but never called `do_sync_after_write()` (then called `sync_on_write()`), so that the file was never synchronized to disk until it was saved. This meant that, for a system whose network disk was stalled for some time, the file could use up system memory on buffered file data.

  We fix this by calling `do_sync_after_write()` each time `NDBFS` writes to a file.

  As part of this work, we increase the autosync size from 1 MB to 16 MB when initializing files.

  Note

  NDB supports `O_SYNC` on platforms that provide it, but does not implement `OM_SYNC` for opening files.

  (Bug #106697, Bug #33946801, Bug #34131456)

Changes in MySQL NDB Cluster 8.0.29 (2022-04-26, General Availability)

MySQL NDB Cluster 8.0.29 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.29 (see Changes in MySQL 8.0.29 (2022-04-26, General Availability)).

Important

This release is no longer available for download. It was removed due to a critical issue that could cause data in `InnoDB` tables having added columns to be interpreted incorrectly. Please upgrade to MySQL Cluster 8.0.30 instead.

- Compilation Notes
- ndbinfo Information Database
- Performance Schema Notes
- Functionality Added or Changed
- Bugs Fixed

Compilation Notes

- `NDB` could not be built using GCC 11 due to an array out of bounds error. (Bug #33459671)
- Removed a number of `-Wstringop-truncation` warnings raised when compiling `NDB` with GCC 9 as well as suppression of such warnings. Also removed unneeded includes from the header file `ndb_global.h`. (Bug #32233543)

ndbinfo Information Database

- Eight new tables providing `NDB` dictionary information about database objects have been added to the `ndbinfo` information database. This makes it possible to obtain a great deal of information of this type by issuing queries in the `mysql` client, without the need to use `ndb_desc`, `ndb_select_all`, and similar utilities. (It is still be necessary to use `ndb_desc` to obtain fragment distribution information.) These tables are listed here, together with the `NDB` objects about which they provide information:
  - `blobs`: Blob tables
  - `dictionary_columns`: Table columns
  - `dictionary_tables`: Tables
  - `events`: Event subscriptions
  - `files`: Files used by disk data tables
  - `foreign_keys`: Foreign keys
  - `hash_maps`: Hash maps
  - `index_columns`: Table indexes

  An additional change in `ndbinfo` is that only `files` and `hash_maps` are defined as views; the remaining six tables listed previously are in fact base tables, even though they are not named using the `ndb$` prefix. As a result, these tables are not hidden as other `ndbinfo` base tables are.

  For more information, see the descriptions of the tables in ndbinfo: The NDB Cluster Information Database. (WL #11968)

Performance Schema Notes

- `ndbcluster` plugin threads can now be seen in the Performance Schema. The `threads` and `setup_threads` tables show all three of these threads: the binary logging thread (`ndb_binlog`

thread), the index statistics thread (`ndb_index_stat` thread), and the metadata thread (`ndb_metadata` thread).

This makes it possible to obtain the thread IDs and thread OS IDs of these threads for use in queries on these and other Performance Schema tables.

For more information and examples, see ndbcluster Plugin Threads. (WL #15000)

Functionality Added or Changed

- **NDB Cluster APIs:** The NDB API now implements a `List::clear()` method which clears all data from a list. This makes it simpler to reuse an existing list with the Dictionary methods `listEvents()`, `listIndexes()`, and `listObjects()`.

  In addition, the `List` destructor has been modified such that it now calls `clear()` before attempting the removal of any elements or attributes from the list being destroyed. (Bug #33676070)

- The client receive thread was enabled only when under high load, where the criterion for determining “high load” was that the number of clients waiting in the poll queue (the receive queue) was greater than `min_active_clients_recv_thread` (default: `8`).

  This was a poor metric for determining high load, since a single client, such as the binary log injector thread handling incoming replication events, could experience high load on its own as well. The same was true of a pushed join query (in which very large batches of incoming `TRANSID_AI` signals are received).

  We change the receive thread such that it now sleeps in the poll queue rather than being deactivated completely, so that it is now always available for handling incoming signals, even when the client is not under high load. (Bug #33752914)

- It is now possible to restore the `ndb_apply_status` table from an `NDB` backup, using `ndb_restore` with the `--with-apply-status` option added in this release. In some cases, this information can be useful in new setting up new replication links.

  `--with-apply-status` restores all rows of the `ndb_apply_status` table except for the row for which the `server_id` value is `0`; use `--restore-epoch` to restore this row.

  To use the `--with-apply-status` option, you must also supply `--restore-data` when invoking `ndb_restore`.

  For more information, see the description of the `--with-apply-status` option in the Reference Manual, as well as ndb_apply_status Table. (Bug #32604161, Bug #33594652)

- Previously, when a user query attempted to open an `NDB` table with a missing (or broken) index, the MySQL server raised `NDB` error `4243 Index not found`. Now when such an attempt is made, it is handled as described here:

  - If the query does not make use of the problematic index, the query succeeds with no errors or warnings.
  - If the query attempts to use the missing or broken index, the query is rejected with a warning from `NDB` (`Index idx is not available in NDB. Use "ALTER TABLE tbl ALTER INDEX idx INVISIBLE" to prevent MySQL from attempting to access it, or use "ndb_restore --rebuild-indexes" to rebuild it`), and an error (`ER_NOT_KEYFILE`).

  The rationale for this change is that constraint violations or missing data sometimes make it impossible to restore an index on an `NDB` table, in which case, running `ndb_restore` with `--disable-indexes` restores the data without the index. With this change, once the data is restored from backup, it is possible to use SQL to fix any corrupt data and rebuild the index. (Bug #28584066, WL #14867)

Bugs Fixed

- **Important Change:** The maximum value supported for the `--ndb-batch-size` server option has been increased from `31536000` to `2147483648` (2 GB). (Bug #21040523)

- **Performance:** When profiling multithreaded data nodes (`ndbmtd`) performing a transaction including a large number of inserts, it was found that more than 50% of CPU time was spent in the internal method `Dblqh::findTransaction()`. It was found that, when there were many operations belonging to uncommitted transactions in the hash list searched by this method, the hash buckets overfilled, the result being that an excessive number of CPU cycles were consumed searching through the hash buckets.

  To address this problem, we fix the number of hash buckets at 4095, and scale the size of a hash bucket relative to the maximum number of operations, so that only relatively few items should now be placed in the same bucket. (Bug #33803541)

  References: See also: Bug #33803487.

- **Performance:** When inserting a great many rows into an empty or small table in the same transaction, the rate at which rows were inserted quickly declined to less than 50% of the initial rate; subsequently, it was found that roughly 50% of all CPU time was spent in `Dbacc::getElement()`, and the root cause identified to be the timing of resizing the structures used for storing elements by `DBACC`, growing with the insertion of more rows in the same transaction, and shrinking following a commit.

  We fix this issue by checking for a need to resize immediately following the insertion or deletion of an element. This also handles the subsequent rejection of an insert. (Bug #33803487)

  References: See also: Bug #33803541.

- **Performance:** A considerable amount of time was being spent searching the event buffer data hash (using the internal method `EventBufData_hash::search()`), due to the following issues:

  - The number of buckets proved to be too low under high load, when the hash bucket list could become very large.

  - The hash buckets were implemented using a linked list. Traversing a long linked list can be highly inefficient.

  We fix these problems by using a vector (`std::vector`) rather than a linked list, and by making the array containing the set of hash buckets expandable. (Bug #33796754)

- **Performance:** The internal function `computeXorChecksum()` was implemented such that great care was taken to aid the compiler in generating optimal code, but it was found that it consumed excessive CPU resources, and did not perform as well as a simpler implementation. This function is now reimplemented with a loop summing up `XOR` results over an array, which appears to result in better optimization with both GCC and Clang compilers. (Bug #33757412)

- **Microsoft Windows:** The `CompressedLCP` data node configuration parameter had no effect on Windows platforms.

  > **Note**
  >
  > When upgrading to this release, Windows users should verify the setting for `CompressedLCP`; if it was previously enabled, you may experience an increase in CPU usage by I/O threads following the upgrade, when under load, when restoring data as part of a node restart, or in both cases. If this behavior is not desired, disable `CompressedLCP`.

  (Bug #33727690)

- **Microsoft Windows:** The internal function `Win32AsyncFile::rmrfReq()` did not always check for both **`ERROR_FILE_NOT_FOUND`** and **`ERROR_PATH_NOT_FOUND`** when either condition was likely. (Bug #33727647)

- **Microsoft Windows:** Corrected several minor issues that occurred with file handling on Windows platforms. (Bug #33727629)

- **NDB Replication:** When performing certain schema operations on an `NDB` table, including those involving a copying `ALTER TABLE`, the epoch column in the `mysql.ndb_apply_status` table on the replica was updated to 0, although this should happen only for transactions originating from storage engines other than `NDBCLUSTER`.

  To fix this, we now update (only) the binary log position when writing a row into `ndb_apply_status` from the same server ID as the previous one, but do not overwrite the current epoch when applying schema operations. (Bug #14139386)

- **NDB Cluster APIs:** Hash key generation using the internal API method `NdbBlob::getBlobKeyHash()` ignored the most significant byte of the key. This unnecessarily caused uneven distribution in the NDB API blob hash list, resulting in a increased need for comparing key values, and thus more CPU usage. (Bug #33803583)

  References: See also: Bug #33783274.

- **NDB Cluster APIs:** Removed an unnecessary assertion that could be hit when iterating through the list returned by `Dictionary::listEvents()`. (Bug #33630835)

- Builds on Ubuntu 21.10 using GCC 11 stopped with `-Werror=maybe-uninitialized`. (Bug #33976268)

- In certain cases, `NDB` did not handle node IDs of data nodes correctly. (Bug #33916404)

- In some cases, `NDB` did not validate all node IDs of data nodes correctly. (Bug #33896409)

- In some cases, array indexes were not handled correctly. (Bug #33896389, Bug #33896399, Bug #33916134)

- In some cases, integers were not handled correctly. (Bug #33896356)

- As part of work done in NDB 8.0.23 to implement the `AutomaticThreadConfig` configuration parameter, the maximum numbers of LQH and TC threads supported by `ndbmtd` were raised from 129 each to 332 and 160, respectively. This adversely affected the performance of `execSEND_PACKED()` methods implemented by several NDB kernel blocks, which complete sending of packed signals when the scheduler is about to suspend execution of the current block thread. This was due to continuing simply to iterate over the arrays of such threads despite the arrays' increased size. We fix this by using a bitmask to track the thread states alongside the full arrays. (Bug #33856371)

- When operating on blob columns, `NDB` must add extra operations to read and write the blob head column and blob part rows. These operations are added to the tail of the transaction's operation list automatically when the transaction is executed.

  To insert a new operation prior to a given operation, it was necessary to traverse the operation list from the beginning until the desired operation was found, with a cost proportional to the length L of the list of preceding operations. This is approximately $L^2 / 2$, increasing as more operations are added to the list; when a large number of operations modifying blobs were defined in a batch, this traversal cost was paid for each operation. This had a noticeable impact on performance when reading and writing blobs.

  We fix this by using list splicing in `NdbTransaction::execute()` to eliminate unnecessary traversals of this sort when defining blob operations. (Bug #33797931)

- The block thread scheduler makes frequent calls to `update_sched_config()` to update its scheduling strategy. That involves checking the fill degree of the job buffer queues used to send signals between the nodes' internal block threads. When these queues are about to fill up, the thread scheduler assigns a smaller value to `max_signals` for the next round, in order to reduce the pressure on the job buffers. When the minimum free threshold has been reached, the scheduler yields the CPU while waiting for the consumer threads to free some job buffer slots.

  The fix in NDB 8.0.18 for a previous issue introduced a mechanism whereby the main thread was allowed to continue executing even when this lower threshold had been reached; in some cases the main thread consumed all job buffers, including those held in reserve, leading to an unplanned shutdown of the data node due to resource exhaustion. (Bug #33792362, Bug #33872577)

  References: This issue is a regression of: Bug #29887068.

- Setting up a cluster with one LDM thread and one query thread using the `ThreadConfig` parameter (for example, `ThreadConfig=ldm={cpubind=1},query={cpubind=2}`) led to unplanned shutdowns of data nodes.

  This was due to internal thread variables being assigned the wrong values when there were no main or request threads explicitly assigned. Now we make sure in such cases that these are assigned the thread number of the first receive thread, as expected. (Bug #33791270)

- `NdbEventBuffer` hash key generation for non-character data reused the same 256 hash keys; in addition, strings of zero length were ignored when calculating hash keys. (Bug #33783274)

- The collection of NDB API statistics based on the `EventBytesRecvdCount` event counter incurred excessive overhead. Now this counter is updated using a value which is aggregated as the event buffer is filled, rather than traversing all of the event buffer data in a separate function call.

  For more information, see NDB API Statistics Counters and Variables. (Bug #33778923)

- The internal method `THRConfig::reorganize_ldm_bindings()` behaved unexpectedly, in some cases changing thread bindings after `AutomaticThreadConfig` had already bound the threads to the correct CPUs. We fix this by removing the method, no longer using it when parsing configuration data or adding threads. (Bug #33764260)

- The receiver thread ID was hard-coded in the internal method `TransporterFacade::raise_thread_prio()` such that it always acted to raise the priority of the receiver thread, even when called from the send thread. (Bug #33752983)

- A fix in NDB 8.0.28 addressed an issue with the code used by various `NDB` components, including `Ndb_index_stat`, that checked whether the data nodes were up and running. In clusters with multiple SQL nodes, this resulted in an increase in the frequency of race conditions between index statistics threads trying to create a table event on the `ndb_index_stat_head` table; that is, it was possible for two SQL nodes to try to create the event at the same time, with the losing SQL node raising Error 746 `Event name already exists`. Due to this error, the binary logging thread ended up waiting for the index statistics thread to signal that its own setup was complete, and so the second SQL node timed out with `Could not create index stat system tables` after `--ndb-wait-setup` seconds. (Bug #33728909)

  References: This issue is a regression of: Bug #32019119.

- On a write error, the message printed by `ndbxfrm` referenced the source file rather than the destination file. (Bug #33727551)

- A complex nested join was rejected with the error `FirstInner/Upper has to be an ancestor or a sibling`, which is thrown by the internal `NdbQueryOperation` interface used to

define a pushed join in the SPJ API, indicating that the join-nest dependencies for the interface were not properly defined.

The query showing the issue had the join nest structure `t2, t1, (t3, (t5, t4))`. Neither of the join conditions on `t5` or `t4` had any references or explicit dependencies on table `t3`, but each had an implicit dependency on `t3` in virtue of being in a nest within the same nest as `t3`.

When preparing a pushed join, `NDB` tracks all required table dependencies between tables and join-nests by adding them to the `m_ancestor` bitmask for each table. For nest level dependencies, they should all be added to the first table in the relevant nest. When the relevant dependencies for a specific table are calculated, they include the set of all tables being explicitly refered in the join condition, plus any implicit dependencies due to the join nests the table is a member of, limited by the uppermost table referred to in the join condition.

For this particular join query we did not properly take into account that there might not be any references to tables in the closest upper nest (the nest starting with `t3`); in such cases we are dependent on all nests up to the nest containing the uppermost table referenced. We fix the issue by introducing a while-loop in which we add ancestor nest dependencies until we reach this uppermost table. (Bug #33670002)

- When the transient memory pool (`TransientPool`) used internally by `NDB` grew above 256 MB, subsequent attempts to shrink the pool caused an error which eventually led to an unplanned shutdown of the data node. (Bug #33647601)
- Check that the connection to `NDB` has been set up before querying about statistics for partitions. (Bug #33643512)
- When the ordered index `PRIMARY` was not created for the `ndb_sql_metadata` table, application of stored grants could not proceed due to the missing index.

  We fix this by protecting creation of utility tables (including ndb_sql_metadata) by wrapping the associated `CREATE TABLE` statement with a schema transaction, thus handling rejection of the statement by rollback. In addition, in the event the newly-created table is not created correctly, it is dropped. These changes avoid leaving behind a table that is only partially created, so that the next attempt to create the utility table starts from the beginning of the process. (Bug #33634453)
- Removed `-Wmaybe-uninitialized` warnings which occurred when compiling NDB Cluster with GCC 11.2. (Bug #33611915)
- `NDB` accepted an arbitrary (and invalid) string of characters following a numeric parameter value in the `config.ini` global configuration. For example, it was possible to use either `OverloadLimit=10 "M12L"` or `OverloadLimit=10 M` (which contains a space) and have it interpreted as `OverloadLimit=10M`.

  It was also possible to use a bare letter suffix in place of an expected numeric value, such as `OverloadLimit=M`, and have it interpreted as zero. This happened as well with an arbitrary string whose first letter was one of the MySQL standard modifiers `K`, `M`, or `G`; thus, `OverloadLimit=MAX_UINT` also had the effect of setting `OverloadLimit` to zero.

  Now, only one of the suffixes `K`, `M`, or `G` is accepted with a numeric parameter value, and it must follow the numeric value immediately, with no intervening whitespace characters or quotation marks. In other words, to set `OverloadLimit` to 10 megabytes, you must use one of `OverloadLimit=10000000`, `OverloadLimit=10M`, or `OverloadLimit=10000K`.

  Note

  To maintain availability, you should check your `config.ini` file for any settings that do not conform to the rule enforced as a result of this change

> and correct them prior to upgrading. Otherwise, the cluster may not be able to start afterwards, until you rectify the issue.

(Bug #33589961)

- Enabling `AutomaticThreadConfig` with fewer than 8 CPUs available led to unplanned shutdowns of data nodes. (Bug #33588734)
- Removed the unused source files `buddy.cpp` and `buddy.hpp` from `storage/ndb/src/common/transporter/`. (Bug #33575155)
- The `NDB` stored grants mechanism now sets the session variable `print_identified_with_as_hex` to `true`, so that password hashes stored in the `ndb_sql_metadata` table are formatted as hexadecimal values rather than being formatted as strings. (Bug #33542052)
- Binary log thread event handling includes optional high-verbosity logging, which, when enabled and the connection to `NDB` lost, produces an excess of log messages like these:

  ```
  datetime 2 [Note] [MY-010866] [Server] NDB Binlog: cluster failure for epoch 55/0.
  datetime 2 [Note] [MY-010866] [Server] NDB Binlog: cluster failure for epoch 55/0.
  ```

  Such repeated log messages, not being of much help in diagnosing errors, have been removed. This leaves a similar log message in such cases, from the handling of schema distribution event operation teardown. (Bug #33492244)
- Historically, a number of different methods have been used to enforce compile-time checks of various interdependencies and assumptions in the `NDB` codebase in a portable way. Since the standard `static_assert()` function is now always available, the `NDB_STATIC_ASSERT` and `STATIC_ASSERT` macros have been replaced with direct usage of `static_assert()`. (Bug #33466577)
- When the internal `AbstractQueryPlan` interface determined the access type to be used for a specific table, it tried to work around an optimizer problem where the `ref` access type was specified for a table and later turned out to be accessible by `eq_ref`. The workaround introduced a new issue by sometimes determining `eq_ref` access for a table actually needing `ref` access; in addition, the prior fix did not take into account `UNIQUE USING HASH` indexes, which need either `eq_ref` or full table scan access, even when the MySQL Optimizer regards it as a `ref` access.

  We fix this by first removing the workaround (which had been made obsolete by the proper fix for the previous issue), and then by introducing the setting of `eq_ref` or `full_table_scan` access for hash indexes. (Bug #33451256)

  References: This issue is a regression of: Bug #28965762.
- When a pushed join is prepared but not executed, the `Ndb_pushed_queries_dropped` status variable is incremented. Now, in addition to this, `NDB` now emits a warning `Prepared pushed join could not be executed...` which is passed to `ER_GET_ERRMSG`. (Bug #33449000)
- The deprecated `-r` option for `ndbd` has been removed. In addition, this change also removes extraneous text from the output of `ndbd --help`. (Bug #33362935)

  References: See also: Bug #31565810.
- `ndb_import` sometimes could not parse correctly a `.csv` file containing Windows/DOS-style (`\r\n`) linefeeds. (Bug #32006725)
- The `ndb_import` tool handled only the hidden primary key which is defined by `NDB` when a table does not have an explicit primary key. This caused an error when inserting a row containing `NULL`

for an auto-increment primary key column, even though the same row was accepted by `LOAD DATA INFILE`.

We fix this by adding support for importing a table with one or more instances of `NULL` in an auto-increment primary key column. This includes a check that a table has no more than one auto-increment column; if this column is nullable, it is redefined by `ndb_import` as `NOT NULL`, and any occurrence of `NULL` in this column is replaced by a generated auto-increment value before inserting the row into `NDB`. (Bug #30799495)

- When a node failure is detected, surviving nodes in the same nodegroup as this node attempt to resend any buffered change data to event subscribers. In cases in which there were no outstanding epoch deliveries, that is, the list of unacknowledged GCIs was empty, the surviving nodes made the incorrect assumption that this list would never be empty. (Bug #30509416)

- When executing a copying `ALTER TABLE` of the parent table for a foreign key and the SQL node terminates prior to completion, there remained an extraneous temporary table with (additional, temporary) foreign keys on all child tables. One consequence of this issue was that it was not possible to restore a backup made using `mysqldump --no-data`.

  To fix this, `NDB` now performs cleanup of temporary tables whenever a `mysqld` process connects (or reconnects) to the cluster. (Bug #24935788, Bug #29892252)

- An unplanned data node shutdown occurred following a bus error on Mac OS X for ARM. We fix this by moving the call to `NdbCondition_Signal()` (in `AsyncIoThread.cpp`) such that it executes prior to `NdbMutex_Unlock()`—that is, into the mutex, so that the condition being signalled is not lost during execution. (Bug #105522, Bug #33559219)

- In `DblqhMain.cpp`, a missing return in the internal `execSCAN_FRAGREQ()` function led to an unplanned shutdown of the data node when inserting a nonfatal error. In addition, the condition `!seize_op_rec(tcConnectptr)` present in the same function was never actually checked. (Bug #105051, Bug #33401830, Bug #33671869)

- It was possible to set any of `MaxNoOfFiredTriggers`, `MaxNoOfLocalScans`, and `MaxNoOfLocalOperations` concurrently with `TransactionMemory`, although this is not allowed.

  In addition, it was not possible to set any of `MaxNoOfConcurrentTransactions`, `MaxNoOfConcurrentOperations`, or `MaxNoOfConcurrentScans` concurrently with `TransactionMemory`, although there is no reason to prevent this.

  In both cases, the concurrent settings behavior now matches the documentation for the `TransactionMemory` parameter. (Bug #102509, Bug #32474988)

- When a redo log part is unable to accept an operation's log entry immediately, the operation (a prepare, commit, or abort) is queued, or (prepare only) optionally aborted. By default operations are queued.

  This mechanism was modified in 8.0.23 as part of decoupling local data managers and redo log parts, and introduced a regression whereby it was possible for queued operations to remain in the queued state until all activity on the log part quiesced. When this occurred, operations could remain queued until `DBTC` declared them timed out, and aborted them. (Bug #102502, Bug #32478380)

Changes in MySQL NDB Cluster 8.0.28 (2022-01-18, General Availability)

MySQL NDB Cluster 8.0.28 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.28 (see Changes in MySQL 8.0.28 (2022-01-18, General Availability)).

- Compilation Notes
- Functionality Added or Changed
- Bugs Fixed

Compilation Notes

- `NDB` did not compile using GCC 11 on Ubuntu 21.10. (Bug #33424843)

Functionality Added or Changed

- Added the `ndbinfo` `index_stats` table, which provides very basic information about `NDB` index statistics. It is intended primarily for use in our internal testing, but may be helpful in conjunction with `ndb_index_stat` and other tools. (Bug #32906654)
- Previously, `ndb_import` always tried to import data into a table whose name was derived from the name of the CSV file being read. This release adds a `--table` option (short form: `-t`) for this program, which overrides this behavior and specifies the name of the target table directly. (Bug #30832382)

Bugs Fixed

- **Important Change:** The deprecated data node option `--connect-delay` has been removed. This option was a synonym for `--connect-retry-delay`, which was not honored in all cases; this issue has been fixed, and the option now works correctly. In addition, the short form `-r` for this option has been deprecated, and you should expect it to be removed in a future release. (Bug #31565810)

  References: See also: Bug #33362935.
- **Microsoft Windows:** On Windows, added missing debug and test suite binaries for MySQL Server (commercial) and MySQL NDB Cluster (commercial and community). (Bug #32713189)
- **NDB Replication:** The `mysqld` option `--slave-skip-errors` can be used to allow the replication applier SQL thread to skip over certain numbered errors automatically. This is not recommended in production because it allows replicas to diverge since whole transactions in the binary log are not applied; for `NDBCLUSTER` with its epoch transactions, this results in entire epochs of changes not being applied, likely leading to inconsistent data.

  Ndb also checks the sequence of epochs applied, and stops the replica applier with an error if there is a sequence problem. Where `--slave-skip-errors` is in use, and an error is skipped, this results in a whole epoch transaction being skipped; this is detected on any subsequent attempt to apply an epoch transaction, which results in the replica applier SQL thread being stopped.

  A new option `--ndb-applier-allow-skip-epoch` is added in this release to allow users to ignore wholly skipped epoch transactions, so that they can use the `--slave-skip-errors` option as with other MySQL storage engines. This is intended for use in testing, and not in a production setting. *Use of these options is entirely at your own risk*.

  When `mysqld` is started with the new option (together with `--slave-skip-errors`), detection of a missing epoch generates a warning, but the replica applier SQL thread continues applying. (Bug #33398973)
- **NDB Replication:** The `log_name` column of the `ndb_apply_status` table was created as `VARBINARY`, despite being defined as `VARCHAR`, using the `latin1` character set, causing hex-decoded output when querying the table using some tools.

We fix this by detecting the faulty column type in `ndb_apply_status` and reinstalling the table definition into the data dictionary while connecting to `NDB`, when `mysqld` checks the layout of this table. (Bug #33380726)

- **NDB Cluster APIs:** Several new basic example C++ NDB API programs have been added to the distribution, under `storage/ndb/ndbapi-examples/ndbapi_basic/` in the source tree. These are shorter and should be easier to understand for newcomers to the NDB API than the existing API examples. They also follow recent C++ standards and practices. These examples have also been added to the NDB API documentation; see Basic NDB API Examples, for more information. (Bug #33378579, Bug #33517296)
- **NDB Cluster APIs:** It is no longer possible to use the `DIVERIFYREQ` signal asynchronously. (Bug #33161562)
- Timing of `wait for scans` log output during online reorganization was not performed correctly. As part of this fix, we change timing to generate one message every 10 seconds rather than scaling indefinitely, so as to supply regular updates. (Bug #35523977)
- Added missing values checks in `ndbd` and `ndbmtd`. (Bug #33661024)
- Online table reorganization increases the number of fragments of a table, and moves rows between them. This is done in the following steps:

  1. Copy rows to new fragments
  2. Update distribution information (hashmap count and total fragments)
  3. Wait for scan activity using old distribution to stop
  4. Delete rows which have moved out of existing partitions
  5. Remove reference to old hashmap
  6. Wait for scan activity started since step 2 to stop

  Due to a counting error, it was possible for the reorganization to hang in step 6; the scan reference count was not decremented, and thus never reached zero as expected. (Bug #33523991)
- A `UNIQUE` index created with `USING HASH` does not support ordered or range access operations, but rather only those operations in which the full key is specified, returning at most a single row. Even so, for such an index on an NDB table, range access was still used on the index. (Bug #33466554, Bug #33474345)
- The same pushed join on `NDB` tables returned an incorrect result when the `batched_key_access` optimizer switch was enabled.

  This issue arose as follows: When the batch key access (BKA) algorithm is used to join two tables, a set of batched keys is first collected from one of the tables; a multirange read (MRR) operation is constructed against the other. A set of bounds (ranges) is specified on the MRR, using the batched keys to construct each bound.

  When result rows are returned it is necessary to identify which range each returned row comes from. This is used to identify the outer table row to perform the BKA join with. When the MRR operation in question was a root of a pushed join operation, `SPJ` was unable to retrieve this identifier (`RANGE_NO`). We fix this by implementing the missing `SPJ` API functionality for returning such a `RANGE_NO` from a pushed join query. (Bug #33416308)
- Each query against the `ndinfo.index_stats` table leaked an `NdbRecord`. We fix this by changing the context so that it owns the `NdbRecord` object which it creates and then to release the `NdbRecord` when going out of scope, and by supporting the creation of one and only one record per context. (Bug #33408123)

- A problem with concurrency occurred when updating cached table statistics with changed rows, when several threads updating same table the threads competed for the `NDB_SHARE` mutex in order to update the cached row count.

  We fix this by reimplementing the storage of changed rows using an atomic counter rather than trying to take the mutex and update the actual shared value, which reduces the need to serialize the threads. In addition, we now append the number of changed rows to the row count only when removing the statistics from the cache and provide a separate mutex protecting only the cached statistics. (Bug #33384978)

  References: See also: Bug #32169848.

- If the schema distribution client detected a timeout before freeing the schema object when the coordinator received the schema event, the coordinator processed the stale schema event instead of returning.

  The coordinator did not know whether a schema distribution timeout was detected by the client, and started processing the schema event as soon as the schema object was valid. To fix this, we indicate the state of the schema object and change its state when the client detects the schema distribution timeout and when the schema event is received by the coordinator, so that both the coordinator and the client are aware of this, and remain synchronized. (Bug #33318201)

- The MySQL Optimizer uses two different methods, `handler::read_cost()` and `Cost_model::page_read_cost()`, to estimate the cost for different access methods, but the cost values returned by these were not always comparable; in some cases this led to the wrong index being chosen and longer execution time for effected queries. To fix this for `NDB`, we override the optimizer's `page_read_cost()` method with one specific to `NDBCLUSTER`. It was also found while working on this issue that the `NDB` handler did not implement the `read_time()` method, used by `read_cost()`; this method is now implemented by `ha_ndbcluster`, and thus the optimizer can now properly take into account the cost difference for `NDB` when using a unique key as opposed to an ordered index (range scan). (Bug #33317872)

- When opening `NDB` tables for queries, the index statistics are retrieved to help the optimizer select the optimal query plan. Each client accessing the stats acquires the global index statistics mutex both before and after accessing the statistics. This causes mutex contention affecting query performance, whether or not there are queries are operating on the same tables, or on different ones.

  We fix this by protecting the count of index statistics references with an atomic counter. The problem was clearly visible when benchmarking with more than 32 clients, when throughput did not increase with additional clients. With this fix, the throughput continues to scale with up to 64 clients. (Bug #33317320)

- In certain cases, an event's category was not properly detected. (Bug #33304814)

- It was not possible to add new data nodes running `ndbd` to an existing cluster with data nodes running `ndbd`. (Bug #33193393)

- For a user granted the `NDB_STORED_USER` privilege, the `password_last_changed` column in the `mysql.user` table was updated each time the SQL node was restarted. (Bug #33172887)

- `DBDICT` did not always perform table name checks correctly. (Bug #33161548)

- Added a number of missing ID and other values checks in `ndbd` and `ndbmtd`. (Bug #33161486, Bug #33162047)

- Added a number of missing ID and other values checks in `ndbd` and `ndbmtd`. (Bug #33161259, Bug #33161362)

- `SET_LOGLEVELORD` signals were not always handled correctly. (Bug #33161246)

- `DUMP 11001` did not always handle all of its arguments correctly. (Bug #33157513)

- File names were not always verified correctly. (Bug #33157475)
- Added a number of missing checks in the data nodes. (Bug #32983723, Bug #33157488, Bug #33161451, Bug #33161477, Bug #33162082)
- Added a number of missing ID and other values checks in `ndbd` and `ndbmtd`. (Bug #32983700, Bug #32893708, Bug #32957478, Bug #32983256, Bug #32983339, Bug #32983489, Bug #32983517, Bug #33157527, Bug #33157531, Bug #33161271, Bug #33161298, Bug #33161314, Bug #33161331, Bug #33161372, Bug #33161462, Bug #33161511, Bug #33161519, Bug #33161537, Bug #33161570, Bug #33162059, Bug #33162065, Bug #33162074, Bug #33162082, Bug #33162092, Bug #33162098, Bug #33304819)
- The management server did not always handle events of the wrong size correctly. (Bug #32957547)
- When `ndb_mgmd` is started without the `--config-file` option, the user is expected to provide the connection string for another management server in the same cluster, so that the management server being started can obtain configuration information from the other. If the host address in the connection string could not be resolved, then the `ndb_mgmd` being started hung indefinitely while trying to establish a connection.

  This issue occurred because a failure to connect was treated as a temporary error, which led to the `ndb_mgmd` retrying the connection, which subsequently failed, and so on, repeatedly. We fix this by treating a failure in host name resolution by `ndb_mgmd` as a permanent error, and immediately exiting. (Bug #32901321)
- The order of parameters used in the argument to `ndb_import --csvopt` is now handled consistently, with the rightmost parameter always taking precedence. This also applies to duplicate instances of a parameter. (Bug #32822757)
- In some cases, issues with the redo log while restoring a backup led to an unplanned shutdown of the data node. To fix this, when the redo log file is not available for writes, we now include the correct wait code and waiting log part in the `CONTINUEB` signal before sending it. (Bug #32733659)

  References: See also: Bug #31585833.
- The binary logging thread sometimes attempted to start before all data nodes were ready, which led to excess logging of unnecessary warnings and errors. (Bug #32019919)
- Instituted a number of value checks in the internal `Ndb_table_guard::getTable()` method. This fixes a known issue in which an SQL node underwent an unplanned shutdown while executing `ALTER TABLE` on an `NDB` table, and potentially additional issues. (Bug #30232826)
- Replaced a misleading error message and otherwise improved the behavior of `ndb_mgmd` when the `HostName` could not be resolved. (Bug #28960182)
- A query used by MySQL Enterprise Monitor to monitor memory use in NDB Cluster became markedly less performant as the number of `NDB` tables increased. We fix this as follows:
  - Row counts for virtual `ndbinfo` tables have been made available to the MySQL optimizer
  - Size estimates are now provided for all `ndbinfo` tables
  - Primary keys have been added to most internal `ndbinfo` tables

  Following these improvements, the performance of queries against `ndbinfo` tables should be comparable to queries against equivalent `MyISAM` tables. (Bug #28658625)
- Following improvements in LDM performance made in NDB 8.0.23, an `UPDATE_FRAG_DIST_KEY_ORD` signal was never sent when needed to a data node using node ID `1`. When running the cluster with 3 or 4 replicas and another node in the same node group restarted, this could result in SQL statements being rejected with error MySQL 1297

ER_GET_TEMPORARY_ERRMSG and, subsequently, SHOW WARNINGS reporting error NDB error 1204.

> **Note**
>
> Prior to upgrading to this release, you can work around the issue by restarting data node 1 whenever any other node in the same node group has been restarted.

(Bug #105098, Bug #33460188)

- Following the rolling restart of a data node performed as part of an upgrade from NDB 7.6 to NDB 8.0, the data node underwent a forced shutdown. We fix this by allowing LQHKEYREQ signals to be sent to both the DBLQH and the DBSPJ kernel blocks. (Bug #105010, Bug #33387443)
- When the AutomaticThreadConfig parameter was enabled, NumCPUs was always shown as 0 in the data node log. In addition, when this parameter is in use, thread CPU bindings are now made correctly, and the data node log shows the actual CPU binding for each thread. (Bug #102503, Bug #32474961)
- ndb_blob_tool --help did not return the expected output. (Bug #98158, Bug #30733508)
- NDB did not close any pending schema transactions when returning an error from internal system table creation and drop functions.

Changes in MySQL NDB Cluster 8.0.27 (2021-10-19, General Availability)

MySQL NDB Cluster 8.0.27 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.27 (see Changes in MySQL 8.0.27 (2021-10-19, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change:** The old MaxAllocate data node configuration parameter has no effect in any current version of NDB. As of this release, it is deprecated and subject to removal in a future release. (Bug #52980, Bug #11760559)
- **NDB Cluster APIs:** Conditions pushed as part of a pushed query can now refer to columns from ancestor tables within the same pushed query.

  For example, given a table created using CREATE TABLE t (x INT PRIMARY KEY, y INT) ENGINE=NDB, the query such as that shown here can now employ condition pushdown:

```
SELECT * FROM t AS a
LEFT JOIN t AS b
ON a.x=0 AND b.y>5,
```

Pushed conditions may include any of the common comparison operators `<`, `<=`, `>`, `>=`, `=`, and `<>`.

Values being compared must be of the same type, including length, precision, and scale.

`NULL` handling is performed according to the comparison semantics specified by the ISO SQL standard; any comparison with `NULL` returns `NULL`.

For more information, see Engine Condition Pushdown Optimization.

As part of this work, the following `NdbInterpretedCode` methods are implemented in the NDB API for comparing column values with values of parameters:

- `branch_col_eq_param()`
- `branch_col_ne_param()`
- `branch_col_lt_param()`
- `branch_col_le_param()`
- `branch_col_gt_param()`
- `branch_col_ge_param()`

In addition, a new `NdbScanFilter::cmp_param()` API method makes it possible to define comparisons between column values and parameter values. (WL #14388)

- In environments that monitor and disconnect idle TCP connections, an idle cluster could suffer from unnecessary data node failures, and the failure of more than one data node could lead to an unplanned shutdown of the cluster.

  To fix this problem, we introduce a new keep-alive signal (`GSN_TRP_KEEP_ALIVE`) that is sent on all connections between data nodes on a regular basis, by default once every 6000 milliseconds (one minute). The length of the interval between these signals can be adjusted by setting the `KeepAliveSendInterval` data node parameter introduced in this release, which can be set to 0 to disable keep-alive signals. You should be aware that `NDB` performs no checking that these signals are received and performs no disconnects on their account (this remains the responsibility of the heartbeat protocol). (Bug #32776593)

- A copying `ALTER TABLE` now checks the source table's fragment commit counts before and after performing the copy. This allows the SQL node executing the `ALTER TABLE` to determine whether there has been any concurrent write activity to the table being altered, and, if so, to terminate the operation, which can help avoid silent loss or corruption of data. When this occurs, the `ALTER TABLE` statement is now rejected with the error `Detected change to data in source table during copying ALTER TABLE. Alter aborted to avoid inconsistency`. (Bug #24511580, Bug #25694856, WL #10540)

- The creation and updating of `NDB` index statistics are now enabled by default. In addition, when restoring metadata, `ndb_restore` now creates the index statistics tables if they do not already exist. (WL #14355)

Bugs Fixed

- **Important Change; NDB Cluster APIs:** Since MySQL 8.0 uses the data dictionary to store table metadata, the following NDB API `Table` methods relating to `.FRM` files are now deprecated:

  - `getFrmData()`
  - `getFrmLength()`
  - `setFrm()`

NDB 8.0 uses `getExtraMetadata()` and `setExtraMetadata()` for reading and writing table metadata stored in the MySQL data dictionary; you should expect the `*Frm*()` methods listed previously to be removed in a future release of NDB Cluster. (Bug #28248575)

- **Important Change:** The default value for each of the two `mysqld` options `--ndb-wait-connected` and `--ndb-wait-setup` has been increased from 30 to 120 seconds. (Bug #32850056)
- **Microsoft Windows:** A number of warnings generated when building NDB Cluster and the NDB utilities with Visual Studio 16.9.5 were removed. (Bug #32881961)
- **Microsoft Windows:** On Windows, it was not possible to start data nodes successfully when the cluster was configured to use more than 64 data nodes. (Bug #104682, Bug #33262452)
- **NDB Cluster APIs:** A number of MGM API functions, including `ndb_mgm_create_logevent_handle()`, did not release memory properly. (Bug #32751506)
- **NDB Cluster APIs:** Trying to create an index using `NdbDictionary` with index statistics enabled and the index statistics tables missing resulted in `NDB` error 723 `No such table existed`, the missing table in this context being one of the statistics tables, which was not readily apparent to the user. Now in such cases, `NDB` instead returns error 4714 `Index stats system tables do not exist`, which is added in this release. (Bug #32739080)
- **NDB Cluster APIs:** The MySQL NoSQL Connector for JavaScript included with NDB Cluster is now built using Node.js version 12.2.6.
- A buffer used in the `SUMA` kernel block did not always accommodate multiple signals. (Bug #33246047)
- In `DbtupBuffer.cpp` the priority level is adjusted to what is currently executing in one path, but it was not used for short signals. This leads to the risk of `TRANSID_AI` signals, `SCAN_FRAGCONF` signals, or both sorts of signals arriving out of order. (Bug #33206293)
- A query executed as a pushed join by the `NDB` storage engine returned fewer rows than expected, under the following conditions:
  - The query contained an `IN` or `EXISTS` subquery executed as a pushed join, using the `firstMatch` algorithm.
  - The subquery itself also contained an outer join using at least 2 tables, at least one of which used the `eq_ref` access type.

  (Bug #33181964)
- Part of the work done in NDB 8.0.23 to add query threads to the `ThreadConfig` parameter included the addition of a `TUX` scan context, used to optimize scans, but in some cases this was not set up correctly following the close of a scan. (Bug #33161080, Bug #32794719)

  References: See also: Bug #33379702.
- An `attribute not found` error was returned on a pushed join in `NDB` when looking up a column to add a linked value.

  The issue was caused by use of the wrong lettercase for the name of the column, and is fixed by insuring that we use the unmodified, original name of the column when performing lookups. (Bug #33104337)
- It was possible in certain cases for an array index to exceed `NO_OF_BUCKETS`. (Bug #33019959)
- Changes in NDB 8.0 resulted in a permanent error (NDB Error 261) being returned when the resources needed by a transaction's operations did not fit within those allocated for the transaction

coordinator, rather than a temporary one (Error 233) as in previous versions. This is significant in NDB Replication, in which a temporary error is retried, but a permanent error is not; a permanent error is suitable when the transaction itself is too large to fit in the transaction coordinator without reconfiguration, but when the transaction cannot fit due to consumption of resources by other transactions, the error should be temporary, as the transaction may be able to fit later, or in some other `TC` instance.

The temporary error returned in such cases (NDB error 233) now has a slightly different meaning; that is, that there is insufficient pooled memory for allocating another operation. (Previously, this error meant that the limit set by `MaxNoOfConcurrentOperations` had been reached.)

Rather than conflate these meanings (dynamic allocation and configured limit), we add a new temporary error (Error 234) which is returned when the configured limit has been reached. See Temporary Resource error, and Application error, for more information about these errors. (Bug #32997832)

References: See also: Bug #33092571.

- Added an `ndbrequire()` in `QMGR` to check whether the node ID received from the `CM_REGREF` signal is less than `MAX_NDB_NODES`. (Bug #32983311)
- A check was reported missing from the code for handling `GET_TABLEID_REQ` signals. To fix this issue, all code relating to all `GET_TABLEID_*` signals has been removed from the `NDB` sources, since these signals are no longer used or supported in NDB Cluster. (Bug #32983249)
- Added an `ndbrequire()` in `QMGR` to ensure that process reports from signal data use appropriate node IDs. (Bug #32983240)
- It was possible in some cases to specify an invalid node type when working with the internal management API. Now the API specifically disallows invalid node types, and defines an “unknown” node type (`NDB_MGM_NODE_TYPE_UNKNOWN`) to cover such cases. (Bug #32957364)
- `NdbReceiver` did not always initialize storage for a MySQL `BIT` column correctly. (Bug #32920099)
- Receiving a spurious schema operation reply from a node not registered as a participant in the current schema operation led to an unplanned shutdown of the SQL node.

  Now in such cases we discard replies from any node not registered as a participant. (Bug #32891206)

  References: See also: Bug #30930132, Bug #32509544.
- The values `true` and `false` for Boolean parameters such as `AutomaticThreadConfig` were not handled correctly when set in a `.cnf` file. (This issue did not affect handling of such values in `.ini` files.) (Bug #32871875)
- Removed unneeded copying of a temporary variable which caused a compiler truncation warning in `storage/ndb/src/common/util/version.cpp`. (Bug #32763321)
- The maximum index size supported by the `NDB` index statistics implementation is 3056 bytes. Attempting to create an index of a larger size when the table held enough data to trigger a statistics update caused `CREATE INDEX` to be rejected with the error `Got error 911 'Index stat scan requested on index with unsupported key size' from NDBCLUSTER`.

  This error originated in the `TUX` kernel block during a scan which caused the schema transaction to fail. This scan is triggered during index creation when the table contains a nonzero number of rows; this also occurs during automatic updates of index statistics or execution of `ANALYZE TABLE`.

  Creating the index as part of `CREATE TABLE` or when the table contained no rows returned no error. No statistics were generated in such situations, while `ANALYZE TABLE` returned an error similar to the one above.

We fix this by allowing the index to be created while returning an appropriate warning from a new check introduced at the handler level. In addition, the TUX scan now handles this situation by suppressing the error, and instead returns success, effectively treating the table as an empty fragment. Otherwise, the behavior in such cases remains unchanged, with a warning returned to the client and no index statistics generated, whether or not the table contains any rows. (Bug #32749829)

References: This issue is a regression of: Bug #28714864.

- A CREATE TABLE statement using ordered indexes returned an error when IndexStatAutoCreate was set to 1 and all SQL nodes had been started with --ndb-index-stat-enable=OFF, due to the fact that, when set to OFF, the option prevented the creation of the index statistics tables. Now these tables are always created at mysqld startup regardless of the value of --ndb-index-stat-enable. (Bug #32649528)

- If an NDB schema operation was lost before the coordinator could process it, the client which logged the operation waited indefinitely for the coordinator to complete or abort it. (Bug #32593352)

  References: See also: Bug #32579148.

- ndb_mgmd now writes a descriptive error message to the cluster log when it is invoked with one or more invalid options. (Bug #32554492)

- An IPv6 address used as part of an NDB connection string and which had only decimal digits following the first colon was incorrectly parsed, and could not be used to connect to the management server. (Bug #32532157)

- Simultaneously creating a user and then granting this user the NDB_STORED_USER privilege on different MySQL servers sometimes caused these servers to hang.

  This was due to the fact that, when the NDB storage engine is enabled, all SQL statements that involve users and grants are evaluated to determine whether they effect any users having the NDB_STORED_USER privilege, after which some statements are ignored, some are distributed to all SQL nodes as statements, and some are distributed to all SQL nodes as requests to read and apply a user privilege snapshot. These snapshots are stored in the mysql.ndb_sql_metadata table. Unlike a statement update, which is limited to one SQL statement, a snapshot update can contain up to seven SQL statements per user. Waiting for any lock in the NDB binary logging thread while managing distributed users could easily lead to a deadlock, when the thread was waiting for an exclusive lock on the local ACL cache.

  We fix this problem by implementing explicit locking around NDB_STORED_USER snapshot updates; snapshot distribution is now performed while holding a global read lock on one row of the ndb_sql_metadata table. (Previously, both statement and snapshot distribution were performed asynchronously, with no locking.) Now, when a thread does not obtain this lock on the first attempt, a warning is raised, and the deadlock prevented.

  For more information, see Privilege Synchronization and NDB_STORED_USER. (Bug #32424653)

  References: See also: Bug #32832676.

- It was not possible to create or update index statistics when the cluster was in single user mode, due to transactions being disallowed from any node other than the designated API node granted access, regardless of type. This prevented the data node responsible for starting transactions relating to index statistics from doing so.

  We address this issue by relaxing the constraint in single user mode and allowing transactions originating from data nodes (but not from other API nodes). (Bug #32407897)

- When starting multiple management nodes, the first such node waits for the others to start before committing the configuration, but this was not explicitly communicated to users. In addition, when

data nodes were started without starting all management nodes, no indication was given to users that its node ID was not allocated since no configuration had yet been committed. Now in such cases, the management node prints a message advising the user that the cluster is configured to use multiple management nodes, and to ensure that all such nodes have been started. (Bug #32339789)

- To handle cases in which a cluster is restarted while the MySQL Server (SQL node) is left running, the index statistics thread is notified when an initial cluster start or restart occurs. The index statistics thread forced the creation of a fresh `Ndb` object and checking of various system objects, which is unnecessary when the MySQL Server is started at the same time as the initial Cluster start which led to the unnecessary re-creation of the `Ndb` object.

  We fix this by restarting only the listener in such cases, rather than forcing the `Ndb` object to be re-created. (Bug #29610555, Bug #33130864)

- Removed extraneous spaces that appeared in some entries written by errors in the node logs. (Bug #29540486)

- `ndb_restore` raised a warning to use `--disable-indexes` when restoring data after the metadata had already been restored with `--disable-indexes`.

  When `--disable-indexes` is used to restore metadata before restoring data, the tables in the target schema have no indexes. We now check when restoring data with this option to ensure that there are no indexes on the target table, and print the warning only if the table already has indexes. (Bug #28749799)

- When restoring of metadata was done using `--disable-indexes`, there was no attempt to create indexes or foreign keys dependent on these indexes, but when `ndb_restore` was used without the option, indexes and foreign keys were created. When `--disable-indexes` was used later while restoring data, `NDB` attempted to drop any indexes created in the previous step, but ignored the failure of a drop index operation due to a dependency on the index of a foreign key which had not been dropped. This led subsequently to problems while rebuilding indexes, when there was an attempt to create foreign keys which already existed.

  We fix `ndb_restore` as follows:

  - When `--disable-indexes` is used, `ndb_restore` now drops any foreign keys restored from the backup.

  - `ndb_restore` now checks for the existence of indexes before attempting to drop them.

  (Bug #26974491)

- The `--ndb-nodegroup-map` option for `ndb_restore` did not function as intended, and code supporting it has been removed. The option now does nothing, and any value set for it is ignored. (Bug #25449055)

- Event buffer status messages shown by the event logger have been improved. Percentages are now displayed only when it makes to do so. In addition, if a maximum size is not defined, the printout shows `max=unlimited`. (Bug #21276857)

- File handles and `FileLogHandler` objects created in `MgmtSrvr::configure_eventlogger` were leaked due to an incomplete destructor for `BufferedLogHandler`. This meant that, each time the cluster configuration changed in a running `ndb_mgmd`, the cluster log was reopened and a file handle leaked, which could lead to issues with test programs and possibly to other problems. (Bug #18192573)

- When `--configdir` was specified as `.`, but with a current working directory other than `DataDir`, the binary configuration was created in `DataDir` and not in the current directory. In addition, `ndb_mgmd` would not start when there was an existing binary configuration in `DataDir`.

  We fix this by having `ndb_mgmd` check the path and refusing to start when a relative path is specified for `--configdir`. (Bug #11755867)

- A memory leak occurred when `NDBCLUSTER` was unable to create a subscription for receiving cluster events. Ownership of the provided event data is supposed to be taken over but actually happened only when creation succeeded, in other cases the provided event data simply being lost. (Bug #102794, Bug #32579459)

- `ndb_mgmd` ignores the `--ndb-connectstring` option if `--config-file` is also specified. Now a warning to this effect is issued, if both options are used. (Bug #102738, Bug #32554759)

- The data node configuration parameters `UndoDataBuffer` and `UndoIndexBuffer` have no effect in any currently supported version of NDB Cluster. Both parameters are now deprecated and the presence of either in the cluster configuration file raises a warning; you should expect them to be removed in a future release. (Bug #84184, Bug #26448357)

- Execution of a bulk `UPDATE` statement using a `LIMIT` clause led to a debug assertion when an error was returned by `NDB`. We fix this by relaxing the assertion for `NDB` tables, since we expect in certain scenarios for an error to be returned at this juncture.

Changes in MySQL NDB Cluster 8.0.26 (2021-07-20, General Availability)

MySQL NDB Cluster 8.0.26 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.26 (see Changes in MySQL 8.0.26 (2021-07-20, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **NDB Cluster APIs:** The version of `Node.js` used by `NDB` has been upgraded to 12.22.1. (Bug #32640847)

- **NDB Cluster APIs:** Added the `NdbScanFilter::setSqlCmpSemantics()` method to the NDB API. Previously, `NdbScanFilter` has always treated `NULL` as equal to itself, so that `NULL == NULL` evaluates as Boolean `TRUE`; this is not in accordance with the SQL standard, which requires that `NULL == NULL` returns `NULL`. The new method makes it possible to override the traditional behavior, and enforce SQL-compliant `NULL` comparisons instead, for the lifetime of a given `NdbScanFilter` instance.

  For more information, see NdbScanFilter::setSqlCmpSemantics(), in the *MySQL NDB Cluster API Developer Guide*. (WL #14476)

- `ndb_restore` now supports conversion between `NULL` and `NOT NULL` columns, as follows:

- To restore a `NULL` column as `NOT NULL`, use the `--lossy-conversions` option. The presence of any `NULL` rows in the column causes `ndb_restore` to raise an and exit.
- To restore a `NOT NULL` column as `NULL`, use the `--promote-attributes` option.

For more information, see the descriptions of the indicated `ndb_restore` options. (Bug #32702637)

- Added the `PreferIPVersion` configuration parameter, which controls the addressing preference of the DNS resolver for IPv4 (`4`) or IPv6 (`6`), with `4` being the default. This parameter must be the same for all TCP connections; for this reason, you should set it only in the `[tcp default]` section of the cluster global configuration file. (Bug #32420615)

Bugs Fixed

- **Packaging:** The `ndb-common` man page was removed, and the information it contained moved to other man pages. (Bug #32799519)
- **Packaging:** The `mysqlbinlog` utility was not included in the NDB Cluster Docker image. (Bug #32795044)
- **NDB Replication:** Use of `NDB_STORED_USER` on an SQL node participating in NDB replication sometimes led to the repeated revocation and restoration of grants. We fix this by ensuring that `log_slave_updates` and the originating `server_id` are handled correctly when performing ACL operations in such cases. (Bug #32832676)
- **NDB Replication:** When starting a cluster with 255 SQL nodes, some `mysqld` processes did not properly initialize schema distribution. This caused binary log setup to fail, with the result that the SQL node never became operational. This error occurred when subscribing to schema changes; setting up the `NdbEventOperation` for the `mysql.ndb_schema` table failed. In addition, a MySQL Server also needs to set up a subscription for the `mysql.ndb_schema_result` table so each MySQL Server requires two resources.

  To fix this problem, the effective default for the `MaxNoOfSubscriptions` configuration parameter is now treated as `2 * MaxNoOfTables + 2 * [number of API nodes]`, rather than as `2 * MaxNoOfTables`. (Bug #32380993)
- **NDB Cluster APIs:** The Node.js adapter did not always handle character set and collation numbers correctly. (Bug #32742521)
- **NDB Cluster APIs:** Added the `NDB_LE_EventBufferStatus3` log event type to `Ndb_logevent_type` in the MGM API. This is an extension of the `NDB_LE_EventBufferStatus` type which handles total, maximum, and allocated bytes as 64-bit values.

  As part of this fix, the maximum value of the `ndb_eventbuffer_max_alloc` server system variable is increased to 9223372036854775807 (2^{63} - 1).

  For more information, see The Ndb_logevent_type Type. (Bug #32381666)
- Conditions which were pushable to the `NDBCLUSTER` engine were not pushed down to the table if it was referred to as part of a view or a table subquery. (Bug #32924533)
- RPM builds of `NDB` for Docker which used dynamic linking did not complete due to the inclusion of the `ndbclient` library by `ndbxfrm`. Now `ndbxfrm` uses the internal `ndbgeneral` and `ndbportlib` libraries instead.

  As part of this fix, `ndb_restore` also now links against `ndbgeneral` and `ndbportlib`. (Bug #32886430)
- `NDB` now uses `std::min()` and `std::max()` in place of its own internal macros for determining the minimum and maximum of two numbers. (Bug #32854692)

- Some error messages printed by `ndb_restore` tried to access transactions that were already closed for error information, resulting in an unplanned exit. (Bug #32815725)
- The error messages for `NDB` errors 418 (`Out of transaction buffers in LQH...`), 419 (`Out of signal memory...`), and 805 (`Out of attrinfo records in tuple manager...`) all referred to increasing `LongSignalMemory`, although there is no configuration parameter by that name. All three of these error messages have been corrected to refer to the `LongMessageBuffer` parameter instead. (Bug #32797240)
- An unsuccessful `CREATE TABLE` of an `NDB` table returns a generic error message (`ERROR HY000: Can't create table 'tbl'`), with additional, more specific error messages often pushed as warnings. To assist users who may not be aware of this and see only the generic message, we have added reminder text regarding the *`SHOW WARNINGS`* statement to the generic error message, to prompt the user to obtain additional information that might help resolve a given issue more quickly. (Bug #32788231)
- An `NDB` error which is not mapped to a MySQL handler error code is typically presented to a MySQL user as error 1296 or 1297, with a message indicating the underlying `NDB` error code; one exception to this behavior is a `COMMIT` error (originating from `ndbcluster_commit()`), for which the usual `NDB` error is 4350 `Transaction already aborted`. MySQL eventually passed this to `strerror()` in the C library, where it was prefixed with `Unknown error` or similar, but the precise format of this prefix varied with platform-specific differences with the version of `libc` being used.

  We fix this by creating both a new handler error `HA_ERR_TX_FAILED`, and a new client error `ER_TRANSACTION_FAILED`, associated with SQL State 25000 `Invalid Transaction State`. (Bug #32763179)

  References: See also: Bug #30264401.
- When started with the `--print-full-config` option, `ndb_mgmd` exited with the error `Address already in use`. This is fixed by skipping free port validation when this option is specified. (Bug #32759903)
- Removed unneeded printouts that were generated in the cluster log when executing queries against the `ndbinfo.cluster_locks` table. (Bug #32747486)
- The `DbUtil` class did not call `mysql_library_end()` when a thread using the MySQL client library had finished doing so, and did not release the thread's local resources by calling `mysql_thread_end()`. (Bug #32730214)
- A memory leak took place in `DbUtil` when running a query for the second time using same `DbUtil` instance; the connection check did not detect the existing MYSQL instance, and replaced it without releasing it. (Bug #32730047)
- Returning an error while determining the number of partitions used by a `NDB` table caused the MySQL server to write `Incorrect information in table.frm file` to its error log, despite the fact that the indicated file did not exist. This also led to problems with flooding of the error log when users attempted to open `NDB` tables while the MySQL server was not actually connected to `NDB`.

  We fix this by changing the function that determines the number of partitions to use the value loaded from the MySQL data dictionary without fetching it from `NDB`, which also saves one round trip when opening a table. For the special case in which the table is opened for upgrade, we fall back to fetching the value from `NDB` in the upgrade code path. (Bug #32713166)
- Using duplicate node IDs with `CREATE NODEGROUP` (for example, `CREATE NODEGROUP 11, 11`) could lead to an unplanned shutdown of the cluster. Now when this command includes duplicate node IDs, it raises an error. (Bug #32701583)
- Improved the performance of queries against the `ndbinfo.cluster_locks` table, which could in some cases run quite slowly. (Bug #32655988)

- Fixed a number of issues found in `ndb_print_backup_file` relating to argument parsing, error reporting, and opening of encrypted files using classes from `ndbxfrm`. (Bug #32583227)
- The directory `unittest/ndb` was generated by the build process even though it is not used. This directory is no longer created when building `NDB`. (Bug #32553339)
- To ensure that the log records kept for the redo log in main memory are written to redo log file within one second, a time supervisor in `DBLQH` acquires a lock on the redo log part prior to the write. A fix for a previous issue caused a continueB signal (introduced as part of that fix) to be sent when the redo log file was not yet opened and ready for the write, then to return without releasing the lock. Now such cases we release the acquired lock before waiting for the redo log file to be open and ready for the write. (Bug #32539348)

  References: This issue is a regression of: Bug #31585833.
- Updating the `Ndb` object used for receiving events from `NDB` in the binary log injector thread with the value for `ndb_eventbuffer_max_alloc` was performed both at the start of each epoch and after having handled one event, when it is sufficient to update the value once per epoch.

  We fix this by not updating from the global value during processing of each event, which reduces the amount of work required during each event processing loop. (Bug #32494904)
- Failure to find all blob parts for a blob column while reading from the event stream was not handled properly, which caused the data in the caller's copy-out buffer to be incomplete, with no error returned to the caller.

  When a user of the event API has been notified that data has been received for a table with blob column, it creates a buffer large enough to hold the entire blob and then calls the function to read the blob column from the event stream. Most blob types are stored as several small parts in `NDB`; to read the blob data for a blob column from the event stream, the buffered event data must be traversed to find the blob parts and to copy each part into the provided buffer. Each piece of buffered event data associated with the blob column is examined to see whether it contains the data for the blob part desired. When a blob part is found, it is copied into the buffer at the original offset provided by the caller.

  The function which finds the blob parts can copy out one or more blob parts at a time. This function is normally called several times while putting the blob parts together—first to find the first blob part, then all the parts in the middle (several at a time), and then the remainder in the last part. When the function does not find all requested blob parts in the buffered event data, this results in an inconsistency which may occur due to any of several different cases—all parts may not have been sent, the received parts may have been stored in the wrong place, there is a problem in the logic putting the blob parts together, or possibly some other issue. The inconsistency is detected by comparing how many blob parts have been found with how many were requested to be copied out this time.

  This problem was noticed while investigating problem with an unplanned SQL node shutdown that could occur while executing some `ALTER TABLE` operations, where a debug-compiled `mysqld` asserted after having printed information about missing blob parts; manual code inspection shows that a release-compiled binary would just return the incomplete buffer to the caller. This problem was also noticed in addressing some previous similar issues.

  We fix this problem by returning an error from `NdbEventOperationImpl::readBlobParts()` whenever requested blob parts cannot be found. Since this is a serious inconsistency, we also extend the printout provided when this problem is detected. A sample of the extended printout is shown here:

```
= print_blob_part_bufs ============================
part_start: 0, part_count: 15
table: { id: 13, version: 2, name: 't1' }
column: { attrid: 1, name: 'b' }
blob parts table: { id: 14, version: 2, name: 'NDB$BLOB_13_1' }
```

```
available buffers: {
[0]*:  part_number: 1, size: 2000, offset: 2000
[1]*:  part_number: 14, size: 2000, offset: 28000
[2]*:  part_number: 7, size: 2000, offset: 14000
[3]*:  part_number: 5, size: 2000, offset: 10000
[4]*:  part_number: 3, size: 2000, offset: 6000
[5]*:  part_number: 0, size: 2000, offset: 0
[6] :  part_number: 15, size: 2000, offset: 30000
[7]*:  part_number: 13, size: 2000, offset: 26000
[8]*:  part_number: 12, size: 2000, offset: 24000
[9]*:  part_number: 11, size: 2000, offset: 22000
[10]*:  part_number: 10, size: 2000, offset: 20000
[11]*:  part_number: 9, size: 2000, offset: 18000
[12]*:  part_number: 8, size: 2000, offset: 16000
[13]*:  part_number: 6, size: 2000, offset: 12000
[14]*:  part_number: 4, size: 2000, offset: 8000
[15]*:  part_number: 2, size: 2000, offset: 4000
}
```

(Bug #32469090)

References: See also: Bug #32405937, Bug #30509284.

- A node was permitted during a restart to participate in a backup before it had completed recovery, instead of being made to wait until its recovery was finished. (Bug #32381165)
- Removed `NDB_WIN32` from the NDB Cluster sources. This define was once intended to demarcate code to be conditionally compiled only for Windows, but had long since been superseded for this purpose by `_WIN32`. (Bug #32380725)
- Running out of disk space while performing an `NDB` backup could lead to an unplanned shutdown of the cluster. (Bug #32367250)
- The index statistics thread relies on the binary log injector thread to inform it about initial system restarts. The index statistics thread then (asynchronously) recycles its `Ndb` object and creates its system tables. Depending on timing, it was possible for the index statistics thread not to be ready to serve requests for a period of time during which `NDB` tables were writable. This also led to issues during the setup of stored grants when the data node parameter `IndexStatAutoCreate` was set to 1.

  We fix this in two ways:

  - Make the sending of the signal to the index statistics thread part of binary log setup so that it is detected in a timely fashion
  - Forcing binary log setup to wait until index statistics functionality has been set up in such cases

  (Bug #32355045)
- It was possible to start `ndb_mgmd` with `NoOfReplicas` set equal to 1 and with more than 72 data nodes defined in the `config.ini` file. Now the management server checks for this condition, and refuses to start if it is found. (Bug #32258207)
- It was possible to start `ndb_mgmd` with an invalid value set in `config.ini` for the `NodeGroup` parameter; subsequently, data node processes using that value were unable to start. Now in such cases, the management server refuses to start, and provides an appropriate error message. (Bug #32210176)
- A statement such `ALTER TABLE t1 ALGORITHM=INPLACE, RENAME COLUMN B to b` that performed an in-place rename of a column changing only the lettercase of its name was successful, but the change was not reflected in the `NDB` dictionary (as shown, for example, in the output of `ndb_desc`). We fix this issue by ensuring that the `NDB` dictionary always matches the lettercase specified in the SQL statement, and that this matches the name as stored in the MySQL data dictionary. (Bug #31958327)

- Event buffer congestion could lead to unplanned shutdown of SQL nodes which were performing binary logging. We fix this by updating the binary logging handler to use `Ndb::pollEvents2()` (rather than the deprecated `pollEvents()` method) to catch and handle such errors properly, instead. (Bug #31926584)
- The `--resume` option for `ndb_import` did not work correctly unless the `--stats` option was also specified. (Bug #31107058)
- Reverted a previous change in the scope of the flags used by `INSERT IGNORE` and other similar SQL statements to inform the handler that duplicate key errors during an insert or update do not stop an ongoing transaction. Now these flags are cleared after every write row event, as before. (Bug #27538524)

  References: See also: Bug #22603412. This issue is a regression of: Bug #20017428.
- `NDBCLUSTER` uses bitmaps of type `MY_BITMAP` for keeping track of which columns are to be used in various contexts. When used in short-lived performance-critical code, these are initialized with a bit buffer whose (fixed) size is defined at compile time. The size of these buffers was calculated in multiple ways, which could lead to copy-paste errors, uncertainty whether the buffer is large enough, and possible allocation of excess space.

  We fix this by implementing an internal `Ndb_bitmap_buf` class that takes the number of bits the buffer should hold as a template argument, and changing all occurrences of static bitmap buffers to instances of `Ndb_bitmap_buf`. This also saves several bytes in the condition pushdown code in which the buffers were too large. (Bug #27150799)
- The `NDB` binlog injector thread now detects errors while handling data change events received from the storage engine. If an error is detected, the thread logs error messages and restarts itself, and as part of the restart an exceptional, incident, or `LOST_EVENTS` entry is written to the binary log. This special entry indicates to a replication applier that the binary log is incomplete. (Bug #18280115, Bug #27150740)
- A `DELETE` statement whose `WHERE` clause referred to a `BLOB` column was not executed correctly. (Bug #13881465)
- Analysis of data node and management node logs was sometimes hampered by the fact that not all log messages included timestamps. This is fixed by replacing a number of different logging functions (`printf`, `fprintf`, `ndbout`, `ndbout_c`, `<<` overloading, and so on) with and standardizing on the existing `EventLogger` mechanism which begins each log message with a timestamp in `YYYY-MM-DD HH:MM:SS` format.

  For more information about NDB Cluster event logs and the log message format, see Event Reports Generated in NDB Cluster. (WL #14311)

  References: See also: Bug #21441915, Bug #30455830.

Changes in MySQL NDB Cluster 8.0.25 (2021-05-11, General Availability)

MySQL NDB Cluster 8.0.25 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.25 (see Changes in MySQL 8.0.25 (2021-05-11, General Availability)).

Packaging Notes

- Binary packages that include `curl` rather than linking to the system `curl` library have been upgraded to use `curl` 7.76.0.

Changes in MySQL NDB Cluster 8.0.24 (2021-04-20, General Availability)

MySQL NDB Cluster 8.0.24 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.24 (see Changes in MySQL 8.0.24 (2021-04-20, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **NDB Cluster APIs:** The version of `Node.js` used by `NDB` has been upgraded to 12.20.1. (Bug #32356419)
- **ndbinfo Information Database:** Added the `dict_obj_tree` table to the `ndbinfo` information database. This table provides information about `NDB` database objects similar to what is shown by the `dict_obj_info` table, but presents it in a hierarchical or tree-like fashion that simplifies seeing relationships between objects such as: tables and indexes; tablespaces and data files; log file groups and undo log files.

  An example of such a view of a table `t1`, having a primary key on column `a` and a unique key on column `b`, is shown here:

  ```
  mysql> SELECT indented_name FROM ndbinfo.dict_obj_tree
      -> WHERE root_name = 'test/def/t1';
  +----------------------------+
  | indented_name              |
  +----------------------------+
  | test/def/t1                |
  |   -> sys/def/13/b          |
  |     -> NDB$INDEX_15_CUSTOM |
  |   -> sys/def/13/b$unique   |
  |     -> NDB$INDEX_16_UI     |
  |   -> sys/def/13/PRIMARY    |
  |     -> NDB$INDEX_14_CUSTOM |
  +----------------------------+
  7 rows in set (0.15 sec)
  ```

  For additional information and examples, see The ndbinfo dict_obj_tree Table. (Bug #32198754)

- **ndbinfo Information Database:** Added the `backup_id` table to the `ndbinfo` information database. This table contains a single column (`id`) and a single row, in which the column value is the backup ID of the most recent backup of the cluster taken with the `ndb_mgm` client. If no NDB backups can be found, the value is 0.

Selecting from this table replaces the process of obtaining this information by using the `ndb_select_all` utility to dump the contents of the internal `SYSTAB_0` table, which is error-prone and can require an excessively long time to complete. (Bug #32073640)

- Added the status variable `Ndb_config_generation`, which shows the generation number of the current configuration being used by the cluster. This can be used as an indicator to determine whether the configuration of the cluster has changed. (Bug #32247424)
- NDB Cluster now uses the MySQL `host_application_signal` component service to perform shutdown of SQL nodes. (Bug #30535835, Bug #32004109)
- `NDB` has implemented the following two improvements in calculation of index statistics:
  - Previously, index statistics were collected from a single fragment only; this is changed such that additional fragments are used for these.
  - The algorithm used for very small tables, such as those having very few rows where results are discarded, has been improved, so that estimates for such tables should be more accurate than previously.

  See NDB API Statistics Counters and Variables, for more information. (WL #13144)
- A number of NDB Cluster programs now support input of the password for encrypting or decrypting an `NDB` backup from standard input. Changes relating to each program affected are listed here:
  - For `ndb_restore`, the `--backup-password-from-stdin` option introduced in this release enables input of the password in a secure fashion, similar to how it is done by the `mysql` client' `--password` option. Use this option together with the `--decrypt` option.
  - `ndb_print_backup_file` now also supports `--backup-password-from-stdin` as the long form of the existing `-P` option.
  - For `ndb_mgm`, `--backup-password-from-stdin` is supported together with `--execute "START BACKUP [options]"` for starting an encrypted cluster backup from the system shell, and has the same effect.
  - Two options for `ndbxfrm`, `--encrypt-password-from-stdin` and `--decrypt-password-from-stdin`, which are also introduced in this release, cause similar behavior when using this program, respectively, to encrypt or to decrypt a backup file.

  In addition, you can cause `ndb_mgm` to use encryption whenever it creates a backup by starting it with `--encrypt-backup`. In this case, the user is prompted for a password when invoking `START BACKUP` if none is supplied. This option can also be specified in the `[ndb_mgm]` section of the `my.cnf` file.

  Also, the behavior and syntax of the `ndb_mgm` management client `START BACKUP` are changed slightly, such that it is now possible to use the `ENCRYPT` option without also specifying `PASSWORD`. Now when the user does this, the management client prompts the user for a password.

  For more information, see the descriptions of the NDB Cluster programs and program options just mentioned, as well as Online Backup of NDB Cluster. (WL #14259)

Bugs Fixed

- **Packaging:** The `mysql-cluster-community-server-debug` and `mysql-cluster-commercial-server-debug` RPM packages were dependent on `mysql-community-server` and `mysql-commercial-server`, respectively, `instead of mysql-cluster-community-server` and `mysql-cluster-commercial-server`. (Bug #32683923)

- **Packaging:** RPM upgrades from NDB 7.6.15 to 8.0.22 did not succeed due to a file having been moved from the `server` RPM to the `client-plugins` RPM. (Bug #32208337)
- **Linux:** On Linux systems, `NDB` interpreted memory sizes obtained from `/proc/meminfo` as being supplied in bytes rather than kilobytes. (Bug #102505, Bug #32474829)
- **Microsoft Windows:** Removed several warnings which were generated when building NDB Cluster on Windows using Microsoft Visual Studio 2019. (Bug #32107056)
- **Microsoft Windows:** `NDB` failed to start correctly on Windows when initializing the `NDB` library with `ndb_init()`, with the error `Failed to find CPU in CPU group`.

  This issue was due to how Windows works with regard to assigning processes to CPUs: when there are more than 64 logical CPUs on a machine, Windows divides them into different processor groups during boot. Each processor group can at most hold 64 CPUs; by default, a process can be assigned to only one processor group. The function `std::thread::hardware_concurrency()` was used to get the maximum number of logical CPUs on the machine, but on Windows, this function returns only the maximum number of logical CPUs present in the processor group with which the current process is affiliated. This value is used to allocate memory for an array that holds hardware information about each CPU on the machine. Since the array held valid memory for CPUs from only one processor group, any attempt to store and retrieve hardware information about a CPU in a different processor group led to array bound read/write errors, leading to memory corruption and ultimately leads to process failures.

  Fixed by using `GetActiveProcessorCount()` instead of the `hardware_concurrency()` function referenced previously. (Bug #101347, Bug #32074703)
- **Solaris:** While preparing `NDBFS` for handling of encrypted backups, activation of `O_DIRECT` was suspended until after initialization of files was completed. This caused initialization of redo log files to require an excessive amount of time on systems using hard disk drives with `ext3` file systems.

  On Solaris, `directio` is used instead of `O_DIRECT`; activating `directio` prior to initialization of files caused a notable increase in time required when using hard disk drives with `UFS` file systems.

  Now we ensure that, on systems having `O_DIRECT`, this is activated before initialization of files, and that, on Solaris, `directio` continues to be activated after initialization of files. (Bug #32187942)
- **NDB Cluster APIs:** Several NDB API coding examples included in the source did not release all resources allocated. (Bug #31987735)
- **NDB Cluster APIs:** Some internal dictionary objects in `NDB` used an internal name format which depends on the database name of the `Ndb` object. This dependency has been made more explicit where necessary and otherwise removed.

  Users of the NDB API should be aware that the `fullyQualified` argument to `Dictionary::listObjects()` still works in such a way that specifying it as `false` causes the objects in the list it returns to use fully qualified names. (Bug #31924949)
- **ndbinfo Information Database:** The system variables `ndbinfo_database` and `ndbinfo_table_prefix` are intended to be read-only. It was found that it was possible to set `mysqld` command-line options corresponding to either or both of these; doing so caused the `ndbinfo` database to malfunction. This fix insures that it is no longer possible to set either of these variables in the `mysql` client or from the command line. (Bug #23583256)
- In some cases, a query affecting a user with the `NDB_STORED_USER` privilege could be printed to the MySQL server log without being rewritten. Now such queries are omitted or rewritten to remove any text following the keyword `IDENTIFIED`. (Bug #32541096)
- The value set for the `SpinMethod` data node configuration parameter was ignored. (Bug #32478388)

- The compile-time debug flag `DEBUG_FRAGMENT_LOCK` was enabled by default. This caused increased resource usage by `DBLQH`, even for release builds.

  This is fixed by disabling `DEBUG_FRAGMENT_LOCK` by default. (Bug #32459625)

- `ndb_mgmd` now exits gracefully in the event of a `SIGTERM` just as it does following a management client `SHUTDOWN` command. (Bug #32446105)

- When started on a port which was already in use, `ndb_mgmd` did not throw any errors since the use of `SO_REUSEADDR` on Windows platforms allowed multiple sockets to bind to the same address and port.

  To take care of this issue, we replace `SO_REUSEADDRPORT` with `SO_EXCLUSIVEADDRUSE`, which prevents re-use of a port that is already in use. (Bug #32433002)

- Encountering an error in detection of an initial system restart of the cluster caused the SQL node to exit prematurely. (Bug #32424580)

- The values reported for the `to` and `from` arguments in `job buffer full` issues were reversed. (Bug #32413686)

- Under some situations, when trying to measure the time of a CPU pause, an elapsed time of zero could result. In addition, computing the average for a very fast spin (for example, 100 loops taking less than 100ns) could zero nanoseconds. In both cases, this caused the spin calibration algorithm throw an arithmetic exception due to division by zero.

  We fix both issues by modifying the algorithm so that it ignores zero values when computing mean spin time. (Bug #32413458)

  References: See also: Bug #32497174.

- Table and database names were not formatted correctly in the messages written to the `mysqld` error log when the internal method `Ndb_rep_tab_reader::scan_candidates()` found ambiguous matches for a given database, table, or server ID in the `ndb_replication` table. (Bug #32393245)

- Some queries with nested pushed joins were not processed correctly. (Bug #32354817)

- When `ndb_mgmd` allocates a node ID, it reads through the configuration to find a suitable ID, causing a mutex to be held while performing hostname lookups. Because network address resolution can require large amounts of time, it is not considered good practice to hold such a mutex or lock while performing network operations.

  This issue is fixed by building a list of configured nodes while holding the mutex, then using the list to perform hostname matching and other logic. (Bug #32294679)

- The schema distribution participant failed to start a global checkpoint after writing a reply to the `ndb_schema_result` table, which caused an unnecessary delay before the coordinator received events from the participant notifying it of the result. (Bug #32284873)

- The global DNS cache used in `ndb_mgmd` caused stale lookups when restarting a node on a new machine with a new IP address, which meant that the node could not allocate a node ID.

  This issue is addressed by the following changes:

  - Node ID allocation no longer depends on `LocalDnsCache`

  - `DnsCache` now uses local scope only

  (Bug #32264914)

- `ndb_restore` generated a core file when started with unknown or invalid arguments. (Bug #32257374)

- Auto-synchronization detected the presence of mock foreign key tables in the NDB dictionary and attempted to re-create them in the MySQL server's data dictionary, although these should remain internal to the NDB Dictionary and not be exposed to the MySQL server. To fix this issue, we now ensure that the NDB Cluster auto-synchronization mechanism ignores any such mock tables. (Bug #32245636)
- Improved resource usage associated with handling of cluster configuration data. (Bug #32224672)
- Removed left-over debugging printouts from `ndb_mgmd` showing a client's version number upon connection. (Bug #32210216)

  References: This issue is a regression of: Bug #30599413.
- The backup abort protocol for handling of node failures did not function correctly for single-threaded data nodes (`ndbd`). (Bug #32207193)
- While retrieving sorted results from a pushed-down join using `ORDER BY` with the `index` access method (and without `filesort`), an SQL node sometimes unexpectedly terminated. (Bug #32203548)
- Logging of redo log initialization showed log part indexes rather than log part numbers. (Bug #32200635)
- Signal data was overwritten (and lost) due to use of extended signal memory as temporary storage. Now in such cases, extended signal memory is not used in this fashion. (Bug #32195561)
- When `ClassicFragmentation = 1`, the default number of partitions per node (shown in `ndb_desc` output as `PartitionCount`) is calculated using the lowest number of LDM threads employed by any single live node, and was done only once, even after data nodes left or joined the cluster, possibly with a new configuration changing the LDM thread count and thus the default partition count. Now in such cases, we make sure the default number of partitions per node is recalculated each time data nodes join or leave the cluster.

  This is not an issue in NDB 8.0.23 and later, when `ClassicFragmentation` is set to 0. (Bug #32183985)
- The internal function `Ndb_ReloadHWInfo()` is responsible for updating hardware information for all the CPUs on the host. For the Linux ARM platform, which does not have Level 3 cache information, this assigned a socket ID for the L3 cache ID but failed to record the value for the global variable `num_shared_l3_caches`, which is needed when creating lists of CPUs connected to a shared L3 cache. (Bug #32180383)
- When trying to run two management nodes on the same host and using the same port number, it was not always obvious to users why they did not start. Now in such cases, in addition to writing a message to the error log, an error message `Same port number is specified for management nodes node_id1 and node_id2 (or) they both are using the default port number on same host host_name` is also written to the console, making the source of the issue more immediately apparent. (Bug #32175157)
- Added a `--cluster-config-suffix` option for `ndb_mgmd` and `ndb_config`, for use in internal testing to override a defaults group suffix. (Bug #32157276)
- The management server returned the wrong status for host name matching when some of the host names in configuration did not resolve and client trying to allocate a node ID connected from the host whose host name resolved to a loopback address with the error `Could not alloc node id at <host>:<port>: Connection with id X done from wrong host ip 127.0.0.1, expected <unresolvable_host> (lookup failed)`.

  This caused the connecting client to fail the node ID allocation.

  This issue is fixed by rewriting the internal match_hostname() function so that it contains all logic for how the requesting client address should match the configured hostnames, and so that it first

checks whether the configured host name can be resolved; if not, it now returns a special error so that the client receives an error indicating that node ID allocation can be retried. The new error is `Could not alloc node id at <host>:<port>: No configured host found of node type <type> for connection from ip 127.0.0.1. Some hostnames are currently unresolvable. Can be retried`. (Bug #32136993)

- The internal function `ndb_socket_create_dual_stack()` did not close a newly created socket when a call to `ndb_setsockopt()` was unsuccessful. (Bug #32105957)
- The local checkpoint (LCP) mechanism was changed in NDB 7.6 such that it also detected idle fragments—that is, fragments which had not changed since the last LCP and thus required no on-disk metadata update. The LCP mechanism could then immediately proceed to handle the next fragment. When there were a great many such idle fragments, the CPU consumption required merely to loop through these became highly significant, causing latency spikes in user transactions.

  A 1 ms delay was already inserted between each such idle fragment being handled. Testing later showed this to be too short an interval, and that we are normally not in as great a hurry to complete these idle fragments as we previously believed.

  This fix extends the idle fragment delay time to 20 ms if there are no redo alerts indicating an urgent need to complete the LCP. In case of a low redo alert state we wait 5 ms instead, and for a higher alert state we fall back to the 1 ms delay. (Bug #32068551)

  References: See also: Bug #31655158, Bug #31613158.
- When an `NDB` table was created, it was invalidated in the global dictionary cache, but this was unnecessary. Furthermore, having a table which exists in the global dictionary cache is actually an advantage for subsequent uses of the new table, since it can be found in the table cache without performing a round trip to `NDB`. (Bug #32047456)
- No clear error message was provided when an `ndb_mgmd` process tried to start using the `PortNumber` of a port that was already in use. (Bug #32045786)
- Two problems occurred when `NDB` closed a table:
  - `NDB` failed to detect when the close was done from `FLUSH TABLES`, which meant that the NDB table definitions in the global dictionary cache were not invalidated.
  - When the close was done by a thread which had not used `NDB` earlier—for example when `FLUSH TABLES` or `RESET MASTER` closed instances of `ha_ndbcluster` held in the table definition cache—a new `Thd_ndb` object was allocated, even though there is a fallback to the global `Ndb` object in case the allocation fails, which never occurs in such cases, so it is less wasteful simply to use the global object already provided.

  (Bug #32018394, Bug #32357856)
- Removed a large number of compiler warnings relating to unused function arguments in `NdbDictionaryImpl`. (Bug #31960757)
- Unnecessary casts were performed when checking internal error codes. (Bug #31930166)
- `NDB` continued to use file system paths for determining the names of tables to open or perform DDL on, in spite of the fact that it longer actually uses files for these operations. This required unnecessary translation between character sets, handling the MySQL-specific file system encoding, and parsing. In addition, results of these operations were stored in buffers of fixed size, each instance of which used several hundred bytes of memory unnecessarily. Since the database and table names to use are already available to `NDB` through other means, this translation could be (and has been) removed in most cases. (Bug #31846478)
- Generation of internal statistics relating to `NDB` object counts was found to lead to an increase in transaction latency at very high rates of transactions per second, brought about by returning an excessive number of freed `NDB` objects. (Bug #31790329)

- NDB behaved unpredictably in response an attempt to change permissions on a distributed user (that is, a user having the NDB_STORED_USER privilege) during a binary log thread shutdown and restart. We address this issue by ensuring that the user gets a clear warning Could not distribute ACL change to other MySQL servers whenever distribution does not succeed. This fix also improves a number of mysqld log messages. (Bug #31680765)

- ndb_restore encountered intermittent errors while replaying backup logs which deleted blob values; this was due to deletion of blob parts when a main table row containing blob one or more values was deleted. This is fixed by modifying ndb_restore to use the asynchronous API for blob deletes, which does not trigger blob part deletes when a blob main table row is deleted (unlike the synchronous API), so that a delete log event for the main table deletes only the row from the main table. (Bug #31546136)

- When a table creation schema transaction is prepared, the table is in TS_CREATING state, and is changed to TS_ACTIVE state when the schema transaction commits on the DBDIH block. In the case where the node acting as DBDIH coordinator fails while the schema transaction is committing, another node starts taking over for the coordinator. The following actions are taken when handling this node failure:

  - DBDICT rolls the table creation schema transaction forward and commits, resulting in the table involved changing to TS_ACTIVE state.

  - DBDIH starts removing the failed node from tables by moving active table replicas on the failed node from a list of stored fragment replicas to another list.

  These actions are performed asynchronously many times, and when interleaving may cause a race condition. As a result, the replica list in which the replica of a failed node resides becomes nondeterministic and may differ between the recovering node (that is, the new coordinator) and other DIH participant nodes. This difference violated a requirement for knowing which list the failed node's replicas can be found during the recovery of the failed node recovery on the other participants.

  To fix this, moving active table replicas now covers not only tables in TS_ACTIVE state, but those in TS_CREATING (prepared) state as well, since the prepared schema transaction is always rolled forward.

  In addition, the state of a table creation schema transaction which is being aborted is now changed from TS_CREATING or TS_IDLE to TS_DROPPING, to avoid any race condition there. (Bug #30521812)

- START BACKUP SNAPSHOTSTART WAIT STARTED could return control to the user prior to the backup's restore point from the user point of view; that is the Backup started notification was sent before waiting for the synchronising global checkpoint (GCP) boundary. This meant that transactions committed after receiving the notification might be included in the restored data.

  To fix this problem, START BACKUP now sends a notification to the client that the backup has been started only after the GCP has truly started. (Bug #29344262)

- Upgrading to NDB Cluster 8.0 from a prior release includes an upgrade in the schema distribution mechanism, as part of which the ndb_schema table is dropped and recreated in a way which causes all MySQL Servers connected to the cluster to restart their binary log injector threads, causing a gap event to be written to the binary log. Since the thread restart happens at the same time on all MySQL Servers, no binary log spans the time during which the schema distribution functionality upgrade was performed, which breaks NDB Cluster Replication.

  This issue is fixed by adding support for gracefully reconstituting the schema distribution tables while allowing the injector thread to continue processing changes from the cluster. This is implemented by handling the DDL event notification for DROP TABLE to turn off support for schema distribution temporarily, and to start regular checks to re-create the tables. When the tables have been successfully created again, the regular checks are turned off and support for schema distribution is turned back on.

NDB also now detects automatically when the ndb_apply_status table has been dropped and re-creates it. The drop and re-creation leaves a gap event in the binary log, which in a replication setup causes the replica MySQL Server to stop applying changes from the source until the replication channel is restarted (see ndb_apply_status Table).

In addition, the minimum version required to perform the schema distribution upgrade is raised to 8.0.24, which prevents automatic triggering of the schema distribution upgrade until all connected API nodes support the new upgrade procedure.

For more information, see NDB Cluster Replication Schema and Tables. (Bug #27697409, Bug #30877233)

References: See also: Bug #30876990.

- Fixed a number of issues uncovered when trying to build NDB with GCC 6. (Bug #25038373)
- Calculation of the redo alert state based on redo log usage was overly aggressive, and thus incorrect, when using more than 1 log part per LDM.

Changes in MySQL NDB Cluster 8.0.23 (2021-01-18, General Availability)

MySQL NDB Cluster 8.0.23 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.23 (see Changes in MySQL 8.0.23 (2021-01-18, General Availability)).

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

Deprecation and Removal Notes

- **Important Change:** As part of the terminology changes begun in MySQL 8.0.21 and NDB 8.0.21, the ndb_slave_conflict_role system variable is now deprecated, and is being replaced with ndb_conflict_role.

  In addition, a number of status variables have been deprecated and are being replaced, as shown in the following table:

  Table 1 Deprecated NDB status variables and their replacements

Deprecated variable	Replacement
Ndb_api_adaptive_send_deferred_count_slave	Ndb_api_adaptive_send_deferred_count_replica
Ndb_api_adaptive_send_forced_count_slave	Ndb_api_adaptive_send_forced_count_replica
Ndb_api_adaptive_send_unforced_count_slave	Ndb_api_adaptive_send_unforced_count_replica
Ndb_api_bytes_received_count_slave	Ndb_api_bytes_received_count_replica
Ndb_api_bytes_sent_count_slave	Ndb_api_bytes_sent_count_replica

Deprecated variable	Replacement
`Ndb_api_pk_op_count_slave`	`Ndb_api_pk_op_count_replica`
`Ndb_api_pruned_scan_count_slave`	`Ndb_api_pruned_scan_count_replica`
`Ndb_api_range_scan_count_slave`	`Ndb_api_range_scan_count_replica`
`Ndb_api_read_row_count_slave`	`Ndb_api_read_row_count_replica`
`Ndb_api_scan_batch_count_slave`	`Ndb_api_scan_batch_count_replica`
`Ndb_api_table_scan_count_slave`	`Ndb_api_table_scan_count_replica`
`Ndb_api_trans_abort_count_slave`	`Ndb_api_trans_abort_count_replica`
`Ndb_api_trans_close_count_slave`	`Ndb_api_trans_close_count_replica`
`Ndb_api_trans_commit_count_slave`	`Ndb_api_trans_commit_count_replica`
`Ndb_api_trans_local_read_row_count_slave`	`Ndb_api_trans_local_read_row_count_replica`
`Ndb_api_trans_start_count_slave`	`Ndb_api_trans_start_count_replica`
`Ndb_api_uk_op_count_slave`	`Ndb_api_uk_op_count_replica`
`Ndb_api_wait_exec_complete_count_slave`	`Ndb_api_wait_exec_complete_count_replica`
`Ndb_api_wait_meta_request_count_slave`	`Ndb_api_wait_meta_request_count_replica`
`Ndb_api_wait_nanos_count_slave`	`Ndb_api_wait_nanos_count_replica`
`Ndb_api_wait_scan_result_count_slave`	`Ndb_api_wait_scan_result_count_replica`
`Ndb_slave_max_replicated_epoch`	`Ndb_replica_max_replicated_epoch`

Also as part of this work, the `ndbinfo.table_distribution_status` table's `tab_copy_status` column values `ADD_TABLE_MASTER` and `ADD_TABLE_SLAVE` are deprecated, and replaced by, respectively, `ADD_TABLE_COORDINATOR` and `ADD_TABLE_PARTICIPANT`.

Finally, the `--help` output of some NDB utility programs such as `ndb_restore` has been updated. (Bug #31571031)

- **NDB Client Programs:** Effective with this release, the MySQL NDB Cluster Auto-Installer (`ndb_setup.py`) has been has been removed from the NDB Cluster binary and source distributions, and is no longer supported. (Bug #32084831)

  References: See also: Bug #31888835.

- **ndbmemcache:** `ndbmemcache`, which was deprecated in the previous release of NDB Cluster, has now been removed from NDB Cluster, and is no longer supported. (Bug #32106576)

Functionality Added or Changed

- As part of work previously done in NDB 8.0, the metadata check performed as part of auto-synchronization between the representation of an `NDB` table in the NDB dictionary and its counterpart in the MySQL data dictionary has been extended to include, in addition to table-level properties, the properties of columns, indexes, and foreign keys. (This check is also made by a debug MySQL server when performing a `CREATE TABLE` statement, and when opening an `NDB` table.)

  As part of this work, any mismatches found between an object's properties in the NDB dictionary and the MySQL data dictionary are now written to the MySQL error log. The error log message includes the name of the property, its value in the NDB dictionary, and its value in the MySQL data dictionary. If the object is a column, index, or foreign key, the object's type is also indicated in the message. (WL #13412)

- The `ThreadConfig` parameter has been extended with two new thread types, query threads and recovery threads, intended for scaleout of LDM threads. The number of query threads must be a multiple of the number of LDM threads, up to a maximum of 3 times the number of LDM threads.

It is also now possible when setting `ThreadConfig` to combine the `main` and `rep` threads into a single thread by setting either or both of these arguments to 0.

When one of these arguments is set to 0 but the other remains set to 1, the resulting combined thread is named `main_rep`. When both are set to 0, they are combined with the `recv` thread (assuming that `recv` to 1), and this combined thread is named `main_rep_recv`. These thread names are those shown when checking the `threads` table in the `ndbinfo` information database.

In addition, the maximums for a number of existing thread types have been increased. The new maximums are: LDM threads: 332; TC threads: 128; receive threads: 64; send threads: 64; main threads: 2. (The maximums for query threads and recovery threads are 332 each.) Maximums for other thread types remain unchanged from previous NDB Cluster releases.

Another change related to this work causes `NDB` to employ mutexes for protecting job buffers when more than 32 block threads are in use. This may cause a slight decrease in performance (roughly 1 to 2 percent), but also results in a decrease in the amount of memory used by very large configurations. For example, a setup with 64 threads which used 2 GB of job buffer memory previously should now require only about 1 GB instead. In our testing this has resulted in an overall improvement (on the order of 5 percent) in the execution of very complex queries.

For more information, see the descriptions of the arguments to the `ThreadConfig` parameter discussed previously, and of the `ndbinfo.threads` table. (WL #12532, WL #13219, WL #13338)

- This release adds the possibility of configuring the threads for multithreaded data nodes (`ndbmtd`) automatically by implementing a new data node configuration parameter `AutomaticThreadConfig`. When set to 1, `NDB` sets up the thread assignments automatically, based on the number of processors available to applications. If the system does not limit the number of processors, you can do this by setting `NumCPUs` to the desired number. Automatic thread configuration makes it unnecessary to set any values for `ThreadConfig` or `MaxNoOfExecutionThreads` in `config.ini`; if `AutomaticThreadConfig` is enabled, settings for either of these parameters are not used.

  As part of this work, a set of tables providing information about hardware and CPU availability and usage by NDB data nodes have been added to the `ndbinfo` information database. These tables, along with a brief description of the information provided by each, are listed here:

  - `cpudata`: CPU usage during the past second
  - `cpudata_1sec`: CPU usage per second over the past 20 seconds
  - `cpudata_20sec`: CPU usage per 20-second interval over the past 400 seconds
  - `cpudata_50ms`: CPU usage per 50-millisecond interval during the past second
  - `cpuinfo`: The CPU on which the data node executes
  - `hwinfo`: The hardware on the host where the data node resides

  Not all of the tables listed are available on all platforms supported by NDB Cluster:

  - The `cpudata`, `cpudata_1sec`, `cpudata_20sec`, and `cpudata_50ms` tables are available only on Linux and Solaris operating systems.
  - The `cpuinfo` table is not available on FreeBSD or macOS.

  (WL #13980)

- Added statistical information in the `DBLQH` block which is employed to track the use of key lookups and scans, as well as tracking queries from `DBTC` and `DBSPJ`. By detecting situations in which the load is high, but in which there is not actually any need to decrease the number of rows scanned

per realtime break, rather than checking the size of job buffer queues to decide how many rows to scan, this makes it possible to scan more rows when there is no CPU congestion. This helps improve performance and realtime behaviour when handling high loads. (WL #14081)

- A new method for handling table partitions and fragments is introduced, such that the number of local data managers (LDMs) for a given data node can determined independently of the number of redo log parts, and that the number of LDMs can now be highly variable. `NDB` employs this method when the `ClassicFragmentation` data node configuration parameter, implemented as part of this work, is set to `false`. When this is done, the number of LDMs is no longer used to determine how many partitions to create for a table per data node; instead, the `PartitionsPerNode` parameter, also introduced in this release, now determines this number, which is now used for calculating how many fragments a table should have.

  When `ClassicFragmentation` has its default value `true`, then the traditional method of using the number of LDMs is used to determine how many fragments a table should have.

  For more information, see Multi-Threading Configuration Parameters (ndbmtd). (WL #13930, WL #14107)

Bugs Fixed

- **macOS:** Removed a number of compiler warnings which occurred when building `NDB` for Mac OS X. (Bug #31726693)
- **Microsoft Windows:** Removed a compiler warning `C4146: unary minus operator applied to unsigned type, result still unsigned` from Visual Studio 2013 found in `storage\ndb\src\kernel\blocks\dbacc\dbaccmain.cpp`. (Bug #23130016)
- **Solaris:** Due to a source-level error, `atomic_swap_32()` was supposed to be specified but was not actually used for Solaris builds of NDB Cluster. (Bug #31765608)
- **NDB Replication:** After issuing `RESET REPLICA ALL / RESET SLAVE ALL`, `NDB` failed to detect that the replica had restarted. (Bug #31515760)
- **NDB Cluster APIs:** Removed redundant usage of `strlen()` in the implementation of `NdbDictionary` and related internal classes in the NDB API. (Bug #100936, Bug #31930362)
- **MySQL NDB ClusterJ:** When a `DomainTypeHandler` was instantiated by a `SessionFactory`, it was stored locally in a static map, `typeToHandlerMap`. If multiple, distinct `SessionFactories` for separate connections to the data nodes were obtained by a ClusterJ application, the static `typeToHandlerMap` would be shared by all those factories. When one of the `SessionFactories` was closed, the connections it created were closed and any tables opened by the connections were cleared from the NDB API global cache. However, the `typeToHandlerMap` was not cleared, and through it the other `SessionFactories` keep accessing the `DomainTypeHandlers` of tables that had already been cleared. These obsolete `DomainTypeHandlers` contained invalid `NdbTable` references and any `ndbapi` calls using those table references ended up with errors.

  This patch fixes the issue by making the `typeToHandlerMap` and the related `proxyInterfacesToDomainClassMap` maps local to a `SessionFactory`, so that they are cleared when the `SessionFactory` is closed. (Bug #31710047)
- **MySQL NDB ClusterJ:** Setting `com.mysql.clusterj.connection.pool.size=0` made connections to an NDB Cluster fail. With this fix, setting `com.mysql.clusterj.connection.pool.size=0` disables connection pooling as expected, so that every request for a `SessionFactory` results in the creation of a new factory and separate connections to the cluster can be created using the same connection string. (Bug #21370745, Bug #31721416)
- When calling `disk_page_abort_prealloc()`, the callback from this internal function is ignored, and so removal of the operation record for the `LQHKEYREQ` signal proceeds without waiting. This left

the table subject to removal before the callback had completed, leading to a failure in PGMAN when the page was retrieved from disk.

To avoid this, we add an extra usage count for the table especially for this page cache miss; this count is decremented as soon as the page cache miss returns. This means that we guarantee that the table is still present when returning from the disk read. (Bug #32146931)

- When a table was created, it was possible for a fragment of the table to be checkpointed too early during the next local checkpoint. This meant that Prepare Phase LCP writes were still being performed when the LCP completed, which could lead to problems with subsequent ALTER TABLE statements on the table just created. Now we wait for any potential Prepare Phase LCP writes to finish before the LCP is considered complete. (Bug #32130918)

- Using the maximum size of an index key supported by index statistics (3056 bytes) caused buffer issues in data nodes. (Bug #32094904)

  References: See also: Bug #25038373.

- NDB now prefers CLOCK_MONOTONIC which on Linux is adjusted by frequency changes but is not updated during suspend. On macOS, NDB instead uses CLOCK_UPTIME_RAW which is the same, except that it is not affected by any adjustments.

  In addition, when intializing NdbCondition the monotonic clock to use is taken directly from NdbTick, rather than re-executing the same preprocessor logic used by NdbTick. (Bug #32073826)

- ndb_restore terminated unexpectedly when run with the --decrypt option on big-endian systems. (Bug #32068854)

- When the data node receive thread found that the job buffer was too full to receive, nothing was done to ensure that, the next time it checked, it resumed receiving from the transporter at the same point at which it stopped previously. (Bug #32046097)

- The metadata check failed during auto-synchronization of tables restored using the ndb_restore tool. This was a timing issue relating to indexes, and was found in the following two scenarios encountered when a table had been selected for auto-synchronization:

  1. When the indexes had not yet been created in the NDB dictionary

  2. When the indexes had been created, but were not yet usable

  (Bug #32004637)

- Optimized sending of packed signals by registering the kernel blocks affected and the sending functions which need to be called for each one in a data structure rather than looking up this information each time. (Bug #31936941)

- When two data definition language statements—one on a database and another on a table in the same schema—were run in parallel, it was possible for a deadlock to occur. The DDL statement affecting the database acquired the global schema lock first, but before it could acquire a metadata lock on the database, the statement affecting the table acquired an intention-exclusive metadata lock on the schema. The table DDL statement was thus waiting for the global schema lock to upgrade its metadata lock on the table to an exclusive lock, while the database DDL statement waited for an exclusive metadata lock on the database, leading to a deadlock.

  A similar type of deadlock involving tablespaces and tables was already known to occur; NDB already detected and resolved that issue. The current fix extends that logic to handle databases and tables as well, to resolve the problem. (Bug #31875229)

- Clang 8 raised a warning due to an uninitialized variable. (Bug #31864792)

- An empty page acquired for an insert did not receive a log sequence number. This is necessary in case the page was used previously and thus required undo log execution before being used again. (Bug #31859717)
- No reason was provided when rejecting an attempt to perform an in-place `ALTER TABLE ... ADD PARTITION` statement on a fully replicated table. (Bug #31809290)
- When the master node had recorded a more recent GCI than a node starting up which had performed an unsuccessful restart, subsequent restarts of the latter could not be performed because it could not restore the stated GCI. (Bug #31804713)
- When using 3 or 4 fragment replicas, it is possible to add more than one node at a time, which means that `DBLQH` and `DBDIH` can have distribution keys based on numbers of fragment replicas that differ by up to 3 (that is, `MAX_REPLICAS` - 1), rather than by only 1. (Bug #31784934)
- It was possible in `DBLQH` for an `ABORT` signal to arrive from `DBTC` before it received an `LQHKEYREF` signal from the next local query handler. Now in such cases, the out-of-order `ABORT` signal is ignored. (Bug #31782578)
- `NDB` did not handle correctly the case when an `ALTER TABLE ... COMMENT="..."` statement did not specify `ALGORITHM=COPY`. (Bug #31776392)
- It was possible in some cases to miss the end point of undo logging for a fragment. (Bug #31774459)
- `ndb_print_sys_file` did not work correctly with version 2 of the `sysfile` format that was introduced in NDB 8.0.18. (Bug #31726653)

  References: See also: Bug #31828452.
- `DBLQH` could not handle the case in which identical operation records having the same transaction ID came from different transaction coordinators. This led to locked rows persisting after a node failure, which kept node recovery from completing. (Bug #31726568)
- It is possible for `DBDIH` to receive a local checkpoint having a given ID to restore while a later LCP is actually used instead, but when performing a partial LCP in such cases, the `DIH` block was not fully synchronized with the ID of the LCP used. (Bug #31726514)
- In most cases, when searching a hash index, the row is used to read the primary key, but when the row has not yet been committed the primary key may be read from the copy row. If the row has been deleted, it can no longer be used to read the primary key. Previously in such cases, the primary key was treated as a NULL, but this could lead to making a comparison using uninitialised data.

  Now when this occurs, the comparison is made only if the row has not been deleted; otherwise the row is checked of among the operations in the serial queue. If no operation has the primary key, then any comparison can be reported as not equal, since no entry in the parallel queue can reinsert the row. This needs to be checked due to the fact that, if an entry in the serial queue is an insert then the primary key from this operation must be identified as such to preclude inserting the same primary key twice. (Bug #31688797)
- As with writing redo log records, when the file currently used for writing global checkpoint records becomes full, writing switches to the next file. This switch is not supposed to occur until the new file is actually ready to receive the records, but no check was made to ensure that this was the case. This could lead to an unplanned data node shutdown restoring data from a backup using `ndb_restore`. (Bug #31585833)
- Release of shared global memory when it is no longer required by the `DBSPJ` block now occurs more quickly than previously. (Bug #31321518)

  References: See also: Bug #31231286.
- Stopping 3 nodes out of 4 in a single node group using `kill -9` caused an unplanned cluster shutdown. To keep this from happening under such conditions, `NDB` now ensures that any node

group that has not had any node failures is viewed by arbitration checks as fully viable. (Bug #31245543)

- Multi-threaded index builds could sometimes attempt to use an internal function disallowed to them. (Bug #30587462)
- While adding new data nodes to the cluster, and while the management node was restarting with an updated configuration file, some data nodes terminated unexpectedly with the error `virtual void TCP_Transporter::resetBuffers(): Assertion `!isConnected()' failed`. (Bug #30088051)
- It was not possible to execute `TRUNCATE TABLE` or `DROP TABLE` for the parent table of a foreign key with `foreign_key_checks` set to 0. (Bug #97501, Bug #30509759)
- Optimized the internal `NdbReceiver::unpackNdbRecord()` method, which is used to convert rows retrieved from the data nodes from packed wire format to the NDB API row format. Prior to the change, roughly 13% of CPU usage for executing a join occurred within this method; this was reduced to approximately 8%. (Bug #95007, Bug #29640755)

Changes in MySQL NDB Cluster 8.0.22 (2020-10-19, General Availability)

MySQL NDB Cluster 8.0.22 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.22 (see Changes in MySQL 8.0.22 (2020-10-19, General Availability)).

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

Backup Notes

- To provide protection against unauthorized recovery of data from backups, this release adds support for `NDB` native encrypted backup using AES-256-CBC. Encrypted backup files are protected by a user-supplied password. `NDB` does not save this password; this needs to be done by the user or application. To create an encrypted backup, use `ENCRYPT PASSWORD=`*`password`* with the `ndb_mgm` client `START BACKUP` command (in addition to any other options which may be required). You can also initiate an encrypted backup in applications by calling the MGM API `ndb_mgm_start_backup4()` function.

  To restore from an encrypted backup, use `ndb_restore` with both of the options `--decrypt` and `--backup-password=`*`password`*. `ndb_print_backup_file` can also read encrypted files using the `-P` option added in this release.

  The encryption password used with this feature can be any string of up to 256 characters from the range of printable ASCII characters other than `!`, `'`, `"`, `$`, `%`, `\`, and `^`. When a password is supplied

for encryption or decryption, it must be quoted using either single or double quotation marks. It is possible to specify an empty password using `''` or `""` but this is not recommended.

You can encrypt existing backup files using the `ndbxfrm` utility which is added to the NDB Cluster distribution in this release; this program can also decrypt encrypted backup files. `ndbxfrm` also compresses and decompresses NDB Cluster backup files. The compression method is the same as used by NDB Cluster for creating compressed backups when `CompressedBackup` is enabled.

It is also possible to require encrypted backups using `RequireEncryptedBackup`. When this parameter is enabled (by setting it equal to 1), the management client rejects any attempt to perform a backup that is not encrypted.

For more information, see Using The NDB Cluster Management Client to Create a Backup, as well as ndbxfrm — Compress, Decompress, Encrypt, and Decrypt Files Created by NDB Cluster. (WL #13474, WL #13499, WL #13548)

Deprecation and Removal Notes

- **NDB Client Programs:** Effective with this release, the MySQL NDB Cluster Auto-Installer (`ndb_setup.py`) has been deprecated and is subject to removal in a future version of NDB Cluster. (Bug #31888835)
- **ndbmemcache:** `ndbmemcache` is deprecated in this release of NDB Cluster, and is scheduled for removal in the next release. (Bug #31876970)

Functionality Added or Changed

- **Important Change:** The `Ndb_metadata_blacklist_size` status variable was renamed as `Ndb_metadata_excluded_count`. (Bug #31465469)
- **Packaging:** Made the following improvements to the `server-minimal` RPM for NDB Cluster and the NDB Cluster Docker image:
  - Added `ndb_import` and other helpful utilities.
  - Included NDB utilities are now linked dynamically.
  - The NDB Cluster Auto-Installer is no longer included.
  - `ndbmemcache` is no longer included.

  (Bug #31838832)
- **NDB Replication:** Batching of updates to rows containing columns of MySQL type `BLOB`, `MEDIUMBLOB`, `LONGBLOB`, `TEXT`, `MEDIUMTEXT`, and `LONGTEXT` (“Blob”) by NDB Cluster. This affects `INSERT`, `UPDATE`, and `DELETE` statements of either of the following types:
  - Statements which modify multiple blob columns in the same row
  - Statements which modify multiple rows containing blob columns in the same statement

  This is accomplished by greatly reducing the number of round trips required between an SQL or other API node and the data nodes in the replica cluster, in some cases by a factor of 10 or more.

  Other SQL statements may also see performance benefits from these improvements. Such statements include `LOAD DATA INFILE` and `CREATE TABLE ... SELECT ...` when acting on tables containing one or more Blob columns. In addition, an `ALTER TABLE ... ENGINE = NDB` statement which changes the storage engine of a table that previously used one other than `NDB` and that contains one or more Blob columns may also execute more efficiently than before this enhancement was implemented.

The performance of some SQL statements which update Blob columns is not noticeably improved by this enhancement, due to the fact that they require scans of table Blob columns, which breaks up batching. Such statements include those of the types listed here:

- A `SELECT` which filters rows by matching on a primary key or unique key column which uses a Blob type
- An `UPDATE` or `DELETE` using a `WHERE` condition which does not depend on a unique value
- A copying `ALTER TABLE` statement on a table which already used the `NDB` storage engine prior to executing the statement

Statements modifying only columns of types `TINYBLOB` or `TINYTEXT` (or both) are not affected by this enhancement.

To take maximum advantage of this improvement, you must enable `slave_allow_batching`. It is also recommended that you increase the values used with the `--ndb-batch-size` and `--ndb-blob-write-batch-bytes` MySQL server options to minimize the number of round trips required by the replica cluster to apply epoch transactions. (Bug #27765184, WL #13043)

- Added the CMake option `NDB_UTILS_LINK_DYNAMIC`, to allow dynamic linking of NDB utilities with `ndbclient`. (Bug #31668306)
- IPv6 addressing is now supported for connections to management and data nodes, including connections between management and data nodes with SQL nodes. For IPv6 addressing to work, the operating platform and network on which the cluster is deployed must support IPv6. Hostname resolution to IPv6 addresses must be provided by the operating platform (this is the same as when using IPv4 addressing).

  Mixing IPv4 and IPv6 addresses in the same cluster is not recommended, but this can be made to work in either of the following cases, provided that `--bind-address` is not used with `ndb_mgmd`:

  - Management node configured with IPv6, data nodes configured with IPv4: This works if the data nodes are started with `--ndb-connectstring` set to the IPv4 address of the management nodes.
  - Management node configured with IPv4, data nodes configured with IPv6: This works if the data nodes are started with `--ndb-connectstring` set to the IPv6 address of the management node.

  When upgrading from an NDB version that does not support IPv6 addressing to a version that does so, it is necessary that the network already support both IPv4 and IPv6. The software upgrade must be performed first; after this, you can update the IPv4 addresses used in the `config.ini` configuration file with the desired IPv6 addresses. Finally, in order for the configuration changes to take effect, perform a system restart of the cluster. (WL #12963)

Bugs Fixed

- **Important Change; NDB Cluster APIs:** The NDB Cluster adapter for Node.js was built against an obsolete version of the runtime. Now it is built using Node.js 12.18.3, and only that version or a later version of Node.js is supported by `NDB`. (Bug #31783049)
- **Important Change:** In order to synchronize excluded metadata objects, it was necessary to correct the underlying issue, if any, and then trigger the synchronization of the objects again. This could be achieved though discovery of individual tables, which does not scale well with an increase in the number of tables and SQL nodes. It could also be done by reconnecting the SQL node to the cluster, but doing so also incurs extra overhead.

  To fix this issue, the list of database objects excluded due to synchronization failure is cleared when `ndb_metadata_sync` is enabled by the user. This makes all such objects eligible for

synchronization in the subsequent detection run, which simplifies retrying the synchronization of all excluded objects.

This fix also removes the validation of objects to be retried which formerly took take place at the beginning of each detection run. Since these objects are of interest only while `ndb_metadata_sync` is enabled, the list of objects to be retried is cleared when this variable is disabled, signalling that all changes have been synchronized. (Bug #31569436)

- **Packaging:** The Dojo library included with NDB Cluster has been upgraded to version 1.15.4. (Bug #31559518)
- **NDB Disk Data:** `ndbmtd` sometimes terminated unexpectedly when it could not complete a lookup for a log file group during a restore operation. (Bug #31284086)
- **NDB Disk Data:** While upgrading a cluster having 3 or 4 replicas after creating sufficient disk data objects to fill up the tablespace, and while performing inserts on the disk data tables, trying to stop some data nodes caused others to exit improperly. (Bug #30922322)
- **NDB Replication:** On Unix-based operating systems, binary logs can be flushed by sending a `SIGHUP` signal to the server, but `NDBCLUSTER` expected one of the SQL statements `FLUSH`, `RESET`, or `SHOW BINLOG EVENTS` only. (Bug #31242689)
- **NDB Cluster APIs:** In certain cases, the `Table::getColumn()` method returned the wrong `Column` object. This could happen when the full name of one table column was a prefix of the name of another, or when the names of two columns had the same hash value. (Bug #31774685)
- **NDB Cluster APIs:** It was possible to make invalid sequences of NDB API method calls using blobs. This was because some method calls implicitly cause transaction execution inline, to deal with blob parts and other issues, which could cause user-defined operations not to be handled correctly due to the use of a method executing operations relating to blobs while there still user-defined blob operations pending. Now in such cases, NDB raises a new error 4558 `Pending blob operations must be executed before this call`. (Bug #27772916)
- `ndb_restore --remap-column` did not handle columns containing `NULL` values correctly. Now any offset specified by the mapping function used with this option is not applied to `NULL`, so that `NULL` is preserved as expected. (Bug #31966676)
- The `ndb_print_backup_file` utility did not respect byte order for row data. This tool now performs byte swapping on row page information to ensure the same results on both big-endian and little-endian platforms. (Bug #31831438)

  References: See also: Bug #32470157.
- In some cases following an upgrade from a version of NDB Cluster previous to 8.0.18 to a later one, writing the `sysfile` (see NDB Cluster Data Node File System Directory) and reading back from it did not work correctly. This could occur when explicit node group assignments to data nodes had been made (using the `NodeGroup` parameter); it was possible for node group assignments to change spontaneously, and even possible for node groups not referenced in the configuration file to be added. This was due to issues with version 2 of the `sysfile` format introduced in NDB 8.0.18. (Bug #31828452, Bug #31820201)

  References: See also: Bug #31726653.
- After encountering the data node in the configuration file which used `NodeGroup=65536`, the management server stopped assigning data nodes lacking an explicit `NodeGroup` setting to node groups. (Bug #31825181)
- Data nodes in certain cases experienced fatal memory corruption in the `PGMAN` kernel block due to an invalid assumption that pages were 32KB aligned, when in fact they are normally aligned to the system page size (4096 or 8192 bytes, depending on platform). (Bug #31768450, Bug #31773234)

- Fixed a misspelled define introduced in NDB 8.0.20 which made an internal function used to control adaptive spinning non-operational. (Bug #31765660)
- When executing undo log records during undo log recovery it was possible when hitting a page cache miss to use the previous undo log record multiple times. (Bug #31750627)
- When an SQL node or cluster shutdown occurred during schema distribution while the coordinator was still waiting for the participants, the schema distribution was aborted halfway but any rows in `ndb_schema_result` related to this schema operation were not cleared. This left open the possibility that these rows might conflict with a future reply from a participant if a DDL operation having the same schema operation ID originated from a client using the same node ID.

  To keep this from happening, we now clear all such rows in `ndb_schema_result` during `NDB` binary log setup. This assures that there are no DDL distributions in progress and any rows remaining in the `ndb_schema_result` table are already obsolete. (Bug #31601674)
- Help output from the MySQL Cluster Auto-Installer displayed incorrect version information. (Bug #31589404)
- In certain rare circumstances, `NDB` missed checking for completion of a local checkpoint, leaving it uncompleted, which meant that subsequent local checkpoints could not be executed. (Bug #31577633)
- A data definition statement can sometimes involve reading or writing of multiple rows (or both) from tables; `NDBCLUSTER` starts an `NdbTransaction` to perform these operations. When such a statement was rolled back, `NDBCLUSTER` attempted to roll back the schema change before rolling back the `NdbTransaction` and closing it; this led to the rollback hanging indefinitely while the cluster waited for the `NdbTransaction` object to close before it was able to roll back the schema change.

  Now in such cases, `NDBCLUSTER` rolls back the schema change only after rolling back and closing any open Ndb`Transaction` associated with the change. (Bug #31546868)
- Adding a new user was not always synchronized correctly to all SQL nodes when the `NDB_STORED_USER` privilege was granted to the new user. (Bug #31486931)
- In some cases, `QMGR` returned conflicting `NDB` engine and MySQL server version information, which could lead to unplanned management node shutdown. (Bug #31471959)
- `SUMA` on a node that is starting up should not send a `DICT_UNLOCK_ORD` signal to the `DICT` block on the master node until both all `SUMA_HANDOVER_REQ` signals sent have had `SUMA_HANDOVER_CONF` signals sent in response, and every switchover bucket set up on receiving a `SUMA_HANDOVER_CONF` has completed switchover. In certain rare cases using `NoOfReplicas` > 2, and in which the delay between global checkpoints was unusually short, it was possible for some switchover buckets to be ready for handover before others, and for handover to proceed even though this was the case. (Bug #31459930)
- Attribute ID mapping needs to be performed when reading data from an `NDB` table using indexes or a primary key whose column order is different than that of the table. For unique indexes, a cached attribute ID map is created when the table is opened, and is then used for each subsequent read, but for primary key reads, the map was built for every read. This is changed so that an attribute ID map for primary key is built and cached when opening the table, and used whenever required for any subsequent reads. (Bug #31452597)

  References: See also: Bug #24444899.
- During different phases of the restore process, `ndb_restore` used different numbers of retries for temporary errors as well as different sleep times between retries. This is fixed by implementing consistent retry counts and sleep times across all restore phases. (Bug #31372923)
- Removed warnings generated when compiling `NDBCLUSTER` with Clang 10. (Bug #31344788)

- The SPJ block contains a load throttling mechanism used when generating LQHKEYREQ signals. When these were generated from parent rows from a scan, and this scan had a bushy topology with multiple children performing key lookups, it was possible to overload the job queues with too many LQHKEYREQ signals, causing node shutdowns due to full job buffers. This problem was originally fixed by Bug #14709490. Further investigation of this issue showed that job buffer full errors could occur even if the SPJ query was not bushy. Due to the increase in the internal batch size for SPJ workers in NDB 7.6.4 as part of work done to implement use of multiple fragments when sending SCAN_FRAGREQ signals to the SPJ block, even a simple query could fill up the job buffers when a relatively small number of such queries were run in parallel.

  To fix this problem, we no longer send any further LQHKEYREQ signals once the number of outstanding signals in a given request exceeds 256. Instead, the parent row from which the LQHKEYREQ is produced is buffered, and the correlation ID of this row is stored in the collection of operations to be resumed later. (Bug #31343524)

  References: This issue is a regression of: Bug #14709490.

- MaxDiskWriteSpeedOwnRestart was not honored as an upper bound for local checkpoint writes during a node restart. (Bug #31337487)

  References: See also: Bug #29943227.

- Under certain rare circumstances, DROP TABLE of an NDB table triggered an assert. (Bug #31336431)

- During a node restart, the SUMA block of the node that is starting must get a copy of the subscriptions (events with subscribers) and subscribers (NdbEventOperation instances which are executing) from a node already running. Before the copy is complete, nodes which are still starting ignore any user-level SUB_START or SUB_STOP requests; after the copy is done, they can participate in such requests. While the copy operation is in progress, user-level SUB_START and SUB_STOP requests are blocked using a DICT lock.

  An issue was found whereby a starting node could participate in SUB_START and SUB_STOP requests after the lock was requested, but before it is granted, which resulted in unsuccessful SUB_START and SUB_STOP requests. This fix ensures that the nodes cannot participate in these requests until after the DICT lock has actually been granted. (Bug #31302657)

- Backups errored out with **FsErrInvalidParameters** when the filesystem was running with O_DIRECT and a data file write was not aligned with the 512-byte block size used by O_DIRECT writes. If the total fragment size in the data file is not aligned with the O_DIRECT block size, NDB pads the last write to the required size, but when there were no fragments to write, BACKUP wrote only the header and footer to the data file. Since the header and footer are less than 512 bytes, leading to the issue with the O_DIRECT write.

  This is fixed by padding out the generic footer to 512 bytes if necessary, using an EMPTY_ENTRY, when closing the data file. (Bug #31180508)

- When employing an execution strategy which requires it to buffer received key rows for later use, DBSPJ now manages the buffer memory allocation tree node by tree node, resulting in a significant drop in CPU usage by the DBSPJ block. (Bug #31174015)

- DBSPJ now uses linear memory instead of segmented memory for storing and handling TRANSID_AI signals, which saves approximately 10% of the CPU previously consumed. Due to this change, it is now possible for DBSPJ to accept TRANSID_AI signals in the short signal format; this is more efficient than the long signal format which requires segmented memory. (Bug #31173582, Bug #31173766)

- Altering the table comment of a fully replicated table using ALGORITHM=INPLACE led to an assertion. (Bug #31139313)

- A local data manager (LDM) has a mechanism for ensuring that a fragment scan does not continue indefinitely when it finds too few rows to fill the available batch size in a reasonable amount of time (such as when a ScanFilter evaluates to false for most of the scanned rows). When this time limit, set in DBLQH as 10 ms, has expired, any rows found up to that point are returned, independent of whether the specified batch size has been filled or not. This acts as a keep-alive mechanism between data and API nodes, as well as to avoid keeping any locks held during the scan for too long.

  A side effect of this is that returning result row batches to the DBSPJ block which are filled well below the expected limit could cause performance issues. This was due not only to poor utilization of the space reserved for batches, requiring more NEXTREQ round trips, but because it also caused DBSPJ internal parallelism statistics to become unreliable.

  Since the DBSPJ block never requests locks when performing scans, overly long locks are not a problem for SPJ requests. Thus it is considered safe to let scans requested by DBSPJ to continue for longer than the 10 ms allowed previously, and the limit set in DBLQH has been increased to 100 ms. (Bug #31124065)

- For a pushed join, the output from EXPLAIN FORMAT=TREE did not indicate whether the table access was an index range scan returning multiple rows, or a single-row lookup on a primary or unique key.

  This fix provides also a minor optimization, such that the handler interface is not accessed more than once in an attempt to return more than a single row if the access type is known to be Unique. (Bug #31123930)

- A previous change (made in NDB 8.0.20) made it possible for a pushed join on tables allowing READ_BACKUP to place two SPJ workers on the data node local to the DBTC block while placing no SPJ workers on some other node; this sometime imbalance is intentional, as the SPJ workload (and possible introduced imbalance) is normally quite low compared to the gains of enabling more local reads of the backup fragments. As an unintended side effect of the same change, these two colocated SPJ workers might scan the same subset of fragments in parallel; this broke an assumption in the DBSPJ block that only a single SPJ worker is instantiated on each data node on which the logic for insuring that each SPJ worker starts its scans from a different fragment depends.

  To fix this problem, the starting fragment for each SPJ worker is now calculated based on the root fragment ID from which the worker starts, which is unique among all SPJ workers even when some of them reside on the same node. (Bug #31113005)

  References: See also: Bug #30639165.

- When upgrading a cluster from NDB 8.0.17 or earlier to 8.0.18 or later, data nodes not yet upgraded could shut down unexpectedly following upgrade of the management server (or management servers) to the new software version. This occurred when a management client STOP command was sent to one or more of the data nodes still running the old version and the new master node (also running the old version of the NDB software) subsequently underwent an unplanned shutdown.

  It was found that this occurred due to setting the signal length and number of signal sections incorrectly when sending a GSN_STOP_REQ—one of a number of signals whose length has been increased in NDB 8.0 as part of work done to support greater numbers of data nodes—to the new master. This happened due to the use of stale data retained from sending a GSN_STOP_REQ to the previous master node. To prevent this from happening, ndb_mgmd now sets the signal length and number of sections explicitly each time, prior to sending a GSN_STOP_REQ signal. (Bug #31019990)

- In some cases, when failures occurred while replaying logs and restoring tuples, ndb_restore terminated instead of returning an error. In addition, the number of retries to be attempted for some operations was determined by hard-coded values. (Bug #30928114)

- During schema distribution, if the client was killed after a DDL operation was already logged in the ndb_schema table, but before the participants could reply, the client simply marked all participants as failed in the NDB_SCHEMA_OBJECT and returned. Since the distribution protocol was already in

progress, the coordinator continued to wait for the participants, received their `ndb_schema_result` insert and processed them; meanwhile, the client was open to send another DDL operation; if one was executed and distribution of it was begun before the coordinator could finish processing the previous schema change, this triggered an assertion there should be only one distribution of a schema operation active at any given time.

In addition, when the client returned having detected a thread being killed, it also released the global schema lock (GSL); this could also lead to undefined issues since the participant could make the changes under the assumption that the GSL was still being held by the coordinator.

In such cases, the client should not return after the DDL operation has been logged in the `ndb_schema` table; from this point, the coordinator has control and the client should wait for it to make a decision. Now the coordinator aborts the distribution only in the event of a server or cluster shutdown, and otherwise waits for all participants either to reply, or to time out and mark the schema operation as completed. (Bug #30684839)

- When, during a restart, a data node received a `GCP_SAVEREQ` signal prior to beginning start phase 9, and thus needed to perform a global checkpoint index write to a local data manager's local checkpoint control file, it did not record information from the `DIH` block originating with the node that sent the signal as part of the data written. This meant that, later in start phase 9, when attempting to send a `GCP_SAVECONF` signal in response to the `GCP_SAVEREQ`, this information was not available, which meant the response could not be sent, resulting in an unplanned shutdown of the data node. (Bug #30187949)

- Setting `EnableRedoControl` to `false` did not fully disable `MaxDiskWriteSpeed`, `MaxDiskWriteSpeedOtherNodeRestart`, and `MaxDiskWriteSpeedOwnRestart` as expected. (Bug #29943227)

  References: See also: Bug #31337487.

- A `BLOB` value is stored by `NDB` in multiple parts; when reading such a value, one read operation is executed per part. If a part is not found, the read fails with a `row not found error`, which indicates a corrupted `BLOB`, since a `BLOB` should never have any missing parts. A problem can arise because this error is reported as the overall result of the read operation, which means that mysqld sees no error and reports zero rows returned.

  This issue is fixed by adding a check specifically for the case in wich a blob part is not found. Now, when this occurs, overwriting the `row not found` error with `corrupted blob`, which causes the originating `SELECT` statement to fail as expected. Users of the NDB API should be aware that, despite this change, the `NdbBlob::getValue()` method continues to report the error as `row not found` in such cases. (Bug #28590428)

- Data nodes did not start when the `RealtimeScheduler` configuration parameter was set to 1. This was due to the fact that index builds during startup are performed by temporarily diverting some I/O threads for use as index building threads, and these threads inherited the realtime properties of the I/O threads. This caused a conflict (treated as a fatal error) when index build thread specifications were checked to ensure that they were not realtime threads. This is fixed by making sure that index build threads are not treated as realtime threads regardless of any settings applying to their host I/O threads, which is as actually intended in their design. (Bug #27533538)

- Using an in-place `ALTER TABLE` to drop an index could lead to the unplanned shutdown of an SQL node. (Bug #24444899)

- As the final step when executing `ALTER TABLE ... ALGORITHM=INPLACE`, `NDBCLUSTER` performed a read of the table metadata from the `NDB` dictionary, requiring an extra round trip between the SQL nodes and data nodes, which unnecessarily both slowed down execution of the statement and provided an avenue for errors which `NDBCLUSTER` was not prepared to handle correctly. This issue is fixed by removing the read of `NDB` table metadata during the final phase of executing an in-place `ALTER TABLE` statement. (Bug #99898, Bug #31497026)

- A memory leak could occur when preparing an `NDB` table for an in-place `ALTER TABLE`. (Bug #99739, Bug #31419144)
- Added the `AllowUnresolvedHostNames` configuration parameter. When set to `true`, this parameter overrides the fatal error normally raised when `ndb_mgmd` cannot connect to a given host name, allowing startup to continue and generating only a warning instead. To be effective, the parameter must be set in the cluster global configuration file's `[tcp default]` section. (WL #13860)

Changes in MySQL NDB Cluster 8.0.21 (2020-07-13, General Availability)

MySQL NDB Cluster 8.0.21 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.21 (see Changes in MySQL 8.0.21 (2020-07-13, General Availability)).

- Packaging Notes
- Functionality Added or Changed
- Bugs Fixed

Packaging Notes

- For Windows, MSI installer packages for NDB Cluster now include a check for the required Visual Studio redistributable package, and produce a message asking the user to install it if it is missing. (Bug #30541398)

Functionality Added or Changed

- **NDB Disk Data:** An initial restart of the cluster now causes the removal of all `NDB` tablespaces and log file groups from the `NDB` dictionary and the MySQL data dictionary. This includes the removal of all data files and undo log files associated with these objects. (Bug #30435378)

  References: See also: Bug #29894166.

- The status variable `Ndb_metadata_blacklist_size` is now deprecated, and is replaced in NDB 8.0.22 by `Ndb_metadata_excluded_count`. (Bug #31465469)
- It now possible to consolidate data from separate instances of NDB Cluster into a single target NDB Cluster when the original datasets all use the same schema. This is supported when using backups created using `START BACKUP` in `ndb_mgm` and restoring them with `ndb_restore`, using the `--remap-column` option implemented in this release (along with `--restore-data` and possibly other options). `--remap-column` can be employed to handle cases of overlapping primary, unique, or both sorts of key values between source clusters, and you need to make sure that they do not overlap in the target cluster. This can also be done to preserve other relationships between tables.

  When used together with `--restore-data`, the new option applies a function to the value of the indicated column. The value set for this option is a string of the format *`db.tbl.col:fn:args`*, whose components are listed here:

  - *`db`*: Database name, after performing any renames.

- *tbl*: Table name.
- *col*: Name of the column to be updated. This column's type must be one of INT or BIGINT, and can optionally be UNSIGNED.
- *fn*: Function name; currently, the only supported name is offset.
- *args*: The size of the offset to be added to the column value by offset. The range of the argument is that of the signed variant of the column's type; thus, negative offsets are supported.

You can use --remap-column for updating multiple columns of the same table and different columns of different tables, as well as combinations of multiple tables and columns. Different offset values can be employed for different columns of the same table.

As part of this work, two new options are also added to ndb_desc in this release:

- --auto-inc (short form -a): Includes the next auto-increment value in the output, if the table has an AUTO_INCREMENT column.
- --context (short form -x): Provides extra information about the table, including the schema, database name, table name, and internal ID.

These options may be useful for obtaining information about NDB tables when planning a merge, particularly in situations where the mysql client may not be readily available.

For more information, see the descriptions for --remap-column, --auto-inc, and --context. (Bug #30383950, WL #11796)

- Detailed real-time information about the state of automatic metadata mismatch detection and synchronization can now be obtained from tables in the MySQL Performance Schema. These two tables are listed here:

  - ndb_sync_pending_objects: Contains information about NDB database objects for which mismatches have been detected between the NDB dictionary and the MySQL data dictionary. It does not include objects which have been excluded from mismatch detection due to permanent errors raised when attempting to synchronize them.
  - ndb_sync_excluded_objects: Contains information about NDB database objects which have been excluded because they cannot be synchronized between the NDB dictionary and the MySQL data dictionary, and thus require manual intervention. These objects are no longer subject to mismatch detection until such intervention has been performed.

  In each of these tables, each row corresponds to a database object, and contains the database object's parent schema (if any), the object's name, and the object's type. Types of objects include schemas, tablespaces, log file groups, and tables. The ndb_sync_excluded_objects table shows in addition to this information the reason for which the object has been excluded.

  Performance Schema NDB Cluster Tables, provides further information about these Performance Schema tables. (Bug #30107543, WL #13712)

- ndb_restore now supports different primary key definitions for source and target tables when restoring from an NDB native backup, using the --allow-pk-changes option introduced in this release. Both increasing and decreasing the number of columns making up the original primary key are supported. This may be useful when it is necessary to accommodate schema version changes while restoring data, or when doing so is more efficient or less time-consuming than performing ALTER TABLE statements involving primary key changes on a great many tables following the restore operation.

  When extending a primary key with additional columns, any columns added must not be nullable, and any values stored in any such columns must not change while the backup is being taken.

Changes in the values of any such column while trying to add it to the table's primary key causes the restore operation to fail. Due to the fact that some applications set the values of all columns when updating a row even if the values of one or more of the columns does not change, it is possible to override this behavior by using the `--ignore-extended-pk-updates` option which is also added in this release. If you do this, care must be taken to insure that such column values do not actually change.

When removing columns from the table's primary key, it is not necessary that the columns dropped from the primary key remain part of the table afterwards.

For more information, see the description of the `--allow-pk-changes` option in the documentation for `ndb_restore`. (Bug #26435136, Bug #30383947, Bug #30634010, WL #10730)

- Added the `--ndb-log-fail-terminate` option for `mysqld`. When used, this causes the SQL node to terminate if it is unable to log all row events. (Bug #21911930)

  References: See also: Bug #30383919.

- When a scalar subquery has no outer references to the table to which the embedding condition is attached, the subquery may be evaluated independent of that table; that is, the subquery is not dependent. `NDB` now attempts to identify and evaluate such a subquery before trying to retrieve any rows from the table to which it is attached, and to use the value thus obtained in a pushed condition, rather than using the subquery which provided the value. (WL #13798)

- In MySQL 8.0.17 and later, the MySQL Optimizer transforms `NOT EXISTS` and `NOT IN` queries into antijoins. `NDB` can now push these down to the data nodes.

  This can be done when there is no unpushed condition on the table, and the query fulfills any other conditions which must be met for an outer join to be pushed down. (WL #13796, WL #13978)

Bugs Fixed

- **Important Change; NDB Disk Data:** An online change of tablespace is not supported for `NDB` tables. Now, for an `NDB` table, the statement `ALTER TABLE ndb_table ... ALGORITHM=INPLACE, TABLESPACE=new_tablespace` is specifically disallowed.

  As part of this fix, the output of the `ndb_desc` utility is improved to include the tablespace name and ID for an `NDB` table which is using one. (Bug #31180526)

- The wrong index was used in the array of indexes while dropping an index. For a table with 64 indexes this caused uninitialized memory to be released. This problem also caused a memory leak when a new index was created at any later time following the drop. (Bug #31408095)

- Removed an unnecessary dependency of `ndb_restore` on the `NDBCLUSTER` plugin. (Bug #31347684)

- Objects for which auto-synchronization fails due to temporary errors, such as failed acquisitions of metadata locks, are simply removed from the list of detected objects, making such objects eligible for detection in later cycles in which the synchronization is retried and hopefully succeeds. This best-effort approach is suitable for the default auto-synchronization behaviour but is not ideal when the using the `ndb_metadata_sync` system variable, which triggers synchronization of all metadata, and when synchronization is complete, is automatically set to false to indicate that this has been done.

  What happened, when a temporary error persisted for a sizable length of time, was that metadata synchronization could take much longer than expected and, in extreme cases, could hang indefinitely, pending user action. One such case occurred when using `ndb_restore` with the `--disable-indexes` option to restore metadata, when the synchronization process entered a vicious cycle of detection and failed synchronization attempts due to the missing indexes until the indexes were rebuilt using `ndb_restore --rebuild-indexes`.

The fix for this issue is, whenever `ndb_metadata_sync` is set to `true`, to exclude an object after synchronization of it fails 10 times with temporary errors by promoting these errors to a permanent error, in order to prevent stalling. This is done by maintaining a list of such objects, this list including a count of the number of times each such object has been retried. Validation of this list is performed during change detection in a similar manner to validation of the exclusion list. (Bug #31341888)

- 32-bit platforms are not supported by NDB 8.0. Beginning with this release, the build process checks the system architecture and aborts if it is not 64-bit. (Bug #31340969)

- Page-oriented allocations on the data nodes are divided into nine resource groups, some having pages dedicated to themselves, and some having pages dedicated to shared global memory which can be allocated by any resource group. To prevent the query memory resource group from depriving other, more important resource groups such as transaction memory of resources, allocations for query memory are performed with low priority and are not allowed to use the last 10% of shared global memory. This change was introduced by poolification work done in NDB 8.0.15.

  Subsequently, it was observed that the calculation for the number of pages of shared global memory kept inaccessible to query memory was correct only when no pages were in use, which is the case when the `LateAlloc` data node parameter is disabled (0).

  This fix corrects that calculation as performed when `LateAlloc` is enabled. (Bug #31328947)

  References: See also: Bug #31231286.

- Multi-threaded restore is able to drive greater cluster load than the previous single-threaded restore, especially while restoring of the data file. To avoid load-related issues, the insert operation parallelism specified for an `ndb_restore` instance is divided equally among the part threads, so that a multithreaded instance has a similar level of parallelism for transactions and operations to a single-threaded instance.

  An error in division caused some part threads to have lower insert operation parallelism than they should have, leading to an slower restore than expected. This fix ensures all part threads in a multi-threaded `ndb_restore` instance get an equal share for parallelism. (Bug #31256989)

- `DUMP 1001` (`DumpPageMemoryOnFail`) now prints out information about the internal state of the data node page memory manager when allocation of pages fails due to resource constraints. (Bug #31231286)

- Statistics generated by `NDB` for use in tracking internal objects allocated and deciding when to release them were not calculated correctly, with the result that the threshold for resource usage was 50% higher than intended. This fix corrects the issue, and should allow for reduced memory usage. (Bug #31127237)

- The Dojo toolkit included with NDB Cluster and used by the Auto-Installer was upgraded to version 1.15.3. (Bug #31029110)

- A packed version 1 configuration file returned by `ndb_mgmd` could contain duplicate entries following an upgrade to NDB 8.0, which made the file incompatible with clients using version 1. This occurs due to the fact that the code for handling backwards compatibility assumed that the entries in each section were already sorted when merging it with the default section. To fix this, we now make sure that this sort is performed prior to merging. (Bug #31020183)

- When executing any of the `SHUTDOWN`, `ALL STOP`, or `ALL RESTART` management commands, it is possible for different nodes to attempt to stop on different global checkpoint index (CGI) boundaries. If they succeed in doing so, then a subsequent system restart is slower than normal because any nodes having an earlier stop GCI must undergo takeover as part of the process. When nodes failing on the first GCI boundary cause surviving nodes to be nonviable, surviving nodes suffer an

arbitration failure; this has the positive effect of causing such nodes to halt at the correct GCI, but can give rise to spurious errors or similar.

To avoid such issues, extra synchronization is now performed during a planned shutdown to reduce the likelihood that different data nodes attempt to shut down at different GCIs as well as the use of unnecessary node takeovers during system restarts. (Bug #31008713)

- During an upgrade, a client could connect to an NDB 8.0 data node without specifying a multiple transporter instance ID, so that this ID defaulted to -1. Due to an assumption that this would occur only in the Node starting state with a single transporter, the node could hang during the restart. (Bug #30899046)

- When an NDB cluster was upgraded from a version that does not support the data dictionary to one that does, any DDL executed on a newer SQL node was not properly distributed to older ones. In addition, newer SDI generated during DDL execution was ignored by any data nodes that had not yet been upgraded. These two issues led to schema states that were not consistent between nodes of different NDB software versions.

  We fix this problem by blocking any DDL affecting NDB data objects while an upgrade from a previous NDB version to a version with data dictionary support is ongoing. (Bug #30877440)

  References: See also: Bug #30184658.

- The `mysql.ndb_schema` table, used internally for schema distribution among SQL nodes, has been modified in NDB 8.0. When a cluster is being upgraded from a older version of NDB, the first SQL node to be upgraded updates the definition of this table to match that used by NDB 8.0 GA releases. (For this purpose, NDB now uses 8.0.21 as the cutoff version.) This is done by dropping the existing table and re-creating it using the newer definition. SQL nodes which have not yet been upgraded receive this `ndb_schema` table drop event and enter read-only mode, becoming writable again only after they are upgraded.

  To keep SQL nodes running older versions of NDB from going into read-only mode, we change the upgrade behavior of `mysqld` such that the `ndb_schema` table definition is updated only if all SQL nodes connected to the cluster are running an 8.0 GA version of NDB and thus having the updated `ndb_schema` table definition. This means that, during an upgrade to the current or any later version, no MySQL Server that is being upgraded updates the `ndb_schema` table if there is at least one SQL node with an older version connected to the cluster. Any SQL node running an older version of NDB remains writable throughout the upgrade process. (Bug #30876990, Bug #31016905)

- `ndb_import` did not handle correctly the case where a CSV parser error occurred in a block of input other than the final block. (Bug #30839144)

- When `mysqld` was upgraded to a version that used a new SDI version, all `NDB` tables become inaccessible. This was because, during an upgrade, synchronization of `NDB` tables relies on deserializing the SDI packed into the NDB Dictionary; if the SDI format was of an version older than that used prior to the upgrade, deserialization could not take place if the format was not the same as that of the new version, which made it impossible to create a table object in the MySQL data dictionary.

  This is fixed by making it possible for `NDB` to bypass the SDI version check in the MySQL server when necessary to perform deserialization as part of an upgrade. (Bug #30789293, Bug #30825260)

- When responding to a `SCANTABREQ`, an API node can provide a distribution key if it knows that the scan should work on only one fragment, in which case the distribution key should be the fragment ID, but in some cases a hash of the partition key was used instead, leading to failures in `DBTC`. (Bug #30774226)

- Several memory leaks found in `ndb_import` have been removed. (Bug #30756434, Bug #30727956)

- The master node in a backup shut down unexpectedly on receiving duplicate replies to a `DEFINE_BACKUP_REQ` signal. These occurred when a data node other than the master errored out during the backup, and the backup master handled the situation by sending itself a `DEFINE_BACKUP_REF` signal on behalf of the missing node, which resulted in two replies being received from the same node (a `CONF` signal from the problem node prior to shutting down and the `REF` signal from the master on behalf of this node), even though the master expected only one reply per node. This scenario was also encountered for `START_BACKUP_REQ` and `STOP_BACKUP_REQ` signals.

  This is fixed in such cases by allowing duplicate replies when the error is the result of an unplanned node shutdown. (Bug #30589827)

- When processing a CSV file, `ndb_import` did not accept trailing field terminators at the ends of lines that were accepted by `mysqlimport`. (Bug #30434663)

- When updating `NDB_TABLE` comment options using `ALTER TABLE`, other options which has been set to non-default values when the table was created but which were not specified in the `ALTER TABLE` statement could be reset to their defaults.

  See Setting NDB Comment Options, for more information. (Bug #30428829)

- Removed a memory leak found in the `ndb_import` utility. (Bug #29820879)

- Incorrect handling of operations on fragment replicas during node restarts could result in a forced shutdown, or in content diverging between fragment replicas, when primary keys with nonbinary (case-sensitive) equality conditions were used. (Bug #98526, Bug #30884622)

Changes in MySQL NDB Cluster 8.0.20 (2020-04-27, General Availability)

MySQL NDB Cluster 8.0.20 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.20 (see Changes in MySQL 8.0.20 (2020-04-27, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change:** It is now possible to divide a backup into slices and to restore these in parallel using two new options implemented for the `ndb_restore` utility, making it possible to employ multiple instances of `ndb_restore` to restore subsets of roughly the same size of the backup in parallel, which should help to reduce the length of time required to restore an NDB Cluster from backup.

  The `--num-slices` options determines the number of slices into which the backup should be divided; `--slice-id` provides the ID of the slice (0 to 1 less than the number of slices) to be restored by `ndb_restore`.

  Up to 1024 slices are supported.

For more information, see the descriptions of the `--num-slices` and `--slice-id` options. (Bug #30383937, WL #10691)

- **Important Change:** To increase the rate at which update operations can be processed, `NDB` now supports and by default makes use of multiple transporters per node group. By default, the number of transporters used by each node group in the cluster is equal to the number of the number of local data management (LDM) threads. While this number should be optimal for most use cases, it can be adjusted by setting the value of the `NodeGroupTransporters` data node configuration parameter which is introduced in this release. The maximum is the greater of the number of LDM threads or the number of TC threads, up to an overall maximum of 32 transporters.

  See Multiple Transporters, for additional information. (WL #12837)

- **NDB Client Programs:** Two options are added for the `ndb_blob_tool` utility, to enable it to detect missing blob parts for which inline parts exist, and to replace these with placeholder blob parts (consisting of space characters) of the correct length. To check whether there are missing blob parts, use the `ndb_blob_tool --check-missing` option. To replace with placeholders any blob parts which are missing, use the program's `--add-missing` option, also added in this release. (Bug #28583971)

- **NDB Client Programs:** Removed a dependency from the `ndb_waiter` and `ndb_show_tables` utility programs on the `NDBT` library. This library, used in `NDB` development for testing, is not required for normal use. The visible effect for users from this change is that these programs no longer print `NDBT_ProgramExit - status` following completion of a run. Applications that depend upon this behavior should be updated to reflect this change when upgrading to this release. (WL #13727, WL #13728)

- **MySQL NDB ClusterJ:** The unused `antlr3` plugin has been removed from the ClusterJ `pom` file. (Bug #29931625)

- **MySQL NDB ClusterJ:** The minimum Java version ClusterJ supports for MySQL NDB Cluster 8.0 is now Java 8. (Bug #29931625)

- **MySQL NDB ClusterJ:** A few Java APIs used by ClusterJ are now deprecated in recent Java versions. These adjustments have been made to ClusterJ:

  - Replaced all `Class.newInstance()` calls with `Class.getDeclaredConstructor().newInstance()` calls. Also updated the exception handling and the test cases wherever required.

  - All the `Number` classes' constructors that instantiate an object from a `String` or a primitive type are deprecated. Replaced all such deprecated instantiation calls with the corresponding `valueOf()` method calls.

  - The `Proxy.getProxyClass()` is now deprecated. The `DomainTypeHandlerImpl` class now directly creates a new instance using the `Proxy.newProxyInstance()` method; all references to the `Proxy` class and its constructors are removed from the `DomainTypeHandlerImpl` class. `SessionFactoryImpl` class now uses the interfaces underlying the proxy object to identify the domain class rather than using the Proxy class. Also updated `DomainTypeHandlerFactoryTest`.

  - The `finalize()` method is now deprecated. This patch does not change the overriding finalize() methods, but just suppresses the warnings on them. This deprecation will be handled separately in a later patch.

  - Updated the CMake configuration to treat deprecation warnings as errors when compiling ClusterJ.

  (Bug #29931625)

- NDB now supports versioning for ndbinfo tables, and maintains the current definitions for its tables internally. At startup, NDB compares its supported ndbinfo version with the version stored in the data dictionary. If the versions differ, NDB drops any old ndbinfo tables and recreates them using the current definitions. (WL #11563)

- Many outer joins and semijoins which previously could not be pushed down to the data nodes can now pushed (see Engine Condition Pushdown Optimization).

  Outer joins which can now be pushed include those which meet the following conditions:

  - There are no unpushed conditions on this table
  - There are no unpushed conditions on other tables in the same join nest, or in upper join nests on which it depends
  - All other tables in the same join nest, or in upper join nests on which it depends are also pushed

  A semijoin using an index scan can now be pushed if it meets the the conditions just noted for a pushed outer join, and it uses the firstMatch strategy. (WL #7636, WL #13576)

  References: See also: Bug #28728603, Bug #28672214, Bug #29296615, Bug #29232744, Bug #29161281, Bug #28728007.

- A new and simplified interface is implemented for enabling and configuring adaptive CPU spin. The SpinMethod data node parameter, added in this release, provides the following four settings:

  - StaticSpinning: Disables adaptive spinning; uses the static spinning employed in previous NDB Cluster releases
  - CostBasedSpinning: Enables adaptive spinning using a cost-based model
  - LatencyOptimisedSpinning: Enables adaptive spinning optimized for latency
  - DatabaseMachineSpinning: Enables adaptive spinning optimized for machines hosting databases, where each thread has its own CPU

  Each of these settings causes the data node to use a set of predetermined values, as needed, for one or more of the spin parameters listed here:

  - SchedulerSpinTimer: The data node configuration parameter of this name.
  - EnableAdaptiveSpinning: Enables or disables adaptive spinning; cannot be set directly in the cluster configuration file, but can be controlled directly using DUMP 104004
  - SetAllowedSpinOverhead: CPU time to allow to gain latency; cannot be set directly in the config.ini file, but possible to change directly, using DUMP 104002

  The presets available from SpinMethod should cover most use cases, but you can fine-tune the adaptive spin behavior using the SchedulerSpinTimer data node configuration parameter and

the DUMP commands just listed, as well as additional DUMP commands in the ndb_mgm cluster management client; see the description of SchedulerSpinTimer for a complete listing.

NDB 8.0.20 also adds a new TCP configuration parameter TcpSpinTime which sets the time to spin for a given TCP connection. This can be used to enable adaptive spinning for any such connections between data nodes, management nodes, and SQL or API nodes.

The ndb_top tool is also enhanced to provide spin time information per thread; this is displayed in green in the terminal window.

For more information, see the descriptions of the SpinMethod and TcpSpinTime configuration parameters, the DUMP commands listed or indicated previously, and the documentation for ndb_top. (WL #12554)

Bugs Fixed

- **Important Change:** When lower_case_table_names was set to 0, issuing a query in which the lettercase of any foreign key names differed from the case with which they were created led to an unplanned shutdown of the cluster. This was due to the fact that mysqld treats foreign key names as case insensitive, even on case-sensitive file systems, whereas the manner in which the NDB dictionary stored foreign key names depended on the value of lower_case_table_names, such that, when this was set to 0, during lookup, NDB expected the lettercase of any foreign key names to match that with which they were created. Foreign key names which differed in lettercase could then not be found in the NDB dictionary, even though it could be found in the MySQL data dictionary, leading to the previously described issue in NDBCLUSTER.

  This issue did not happen when lower_case_table_names was set to 1 or 2.

  The problem is fixed by making foreign key names case insensitive and removing the dependency on lower_case_table_names. This means that the following two items are now always true:

  1. Foreign key names are now stored using the same lettercase with which they are created, without regard to the value of lower_case_table_names.
  2. Lookups for foreign key names by NDB are now always case insensitive.

  (Bug #30512043)

- **Packaging:** Removed an unnecessary dependency on Perl from the mysql-cluster-community-server-minimal RPM package. (Bug #30677589)

- **Packaging:** NDB did not compile successfully on Ubuntu 16.04 with GCC 5.4 due to the use of isnan() rather than std::isnan(). (Bug #30396292)

  References: This issue is a regression of: Bug #30338980.

- **OS X:** Removed the variable SCHEMA_UUID_VALUE_LENGTH which was used only once in the NDB sources, and which caused compilation warnings when building on Mac OSX. The variable has been replaced with UUID_LENGTH. (Bug #30622139)

- **NDB Disk Data:** Allocation of extents in tablespace data files is now performed in round-robin fashion among all data files used by the tablespace. This should provide more even distribution of data in cases where multiple storage devices are used for Disk Data storage. (Bug #30739018)

- **NDB Disk Data:** Under certain conditions, checkpointing of Disk Data tables could not be completed, leading to an unplanned data node shutdown. (Bug #30728270)

- **NDB Disk Data:** An uninitialized variable led to issues when performing Disk Data DDL operations following a restart of the cluster. (Bug #30592528)

- **MySQL NDB ClusterJ:** When a Date value was read from a NDB cluster, ClusterJ sometimes extracted the wrong year value from the row. It was because the Utility class, when unpacking

the `Date` value, wrongly extracted some extra bits for the year. This patch makes ClusterJ only extract the required bits. (Bug #30600320)

- **MySQL NDB ClusterJ:** When the cluster's `NdbOperation::AbortOption` type had the value of `AO_IgnoreOnError`, when there was a read error, ClusterJ took that as the row was missing and returned `null` instead of an exception. This was because with `AO_IgnoreOnErro`, the `execute()` method always returns a success code after each transaction, and ClusterJ is supposed to check for any errors in any of the individual operations; however, read operations were not checked by ClusterJ in the case. With this patch, read operations are now checked for errors after query executions, so that a reading error is reported as such. (Bug #30076276)

- The fix for a previous issue in the MySQL Optimizer adversely affected engine condition pushdown for the `NDB` storage engine. (Bug #303756135)

  References: This issue is a regression of: Bug #97552, Bug #30520749.

- When restoring signed auto-increment columns, `ndb_restore` incorrectly handled negative values when determining the maximum value included in the data. (Bug #30928710)

- On an SQL node which had been started with `--ndbcluster`, before any other nodes in the cluster were started, table creation succeeded while creating the `ndbinfo` schema, but creation of views did not, raising `HA_ERR_NO_CONNECTION` instead. (Bug #30846678)

- Formerly (prior to NDB 7.6.4) an `SPJ` worker instance was activated for each fragment of the root table of the pushed join, but in NDB 7.6 and later, a single worker is activated for each data node and is responsible for all fragments on that data node.

  Before this change was made, it was sufficient for each such worker to scan a fragment with parallelism equal to 1 for all `SPJ` workers to keep all local data manager threads busy. When the number of workers was reduced as result of the change, the minimum parallelism should have been increased to equal the number of fragments per worker to maintain the degree of parallelism.

  This fix ensures that this is now done. (Bug #30639503)

- The `ndb_metadata_sync` system variable is set to true to trigger synchronization of metadata between the MySQL data dictionary and the `NDB` dictionary; when synchronization is complete, the variable is automatically reset to false to indicate that this has been done. One scenario involving the detection of a schema not present in the MySQL data dictionary but in use by the `NDB` Dictionary sometimes led to `ndb_metadata_sync` being reset before all tables belonging to this schema were successfully synchronized. (Bug #30627292)

- When using shared user and grants, all `ALTER USER` statements were distributed as snapshots, whether they contained plaintext passwords or not.

  In addition, `SHOW CREATE USER` did not include resource limits (such as `MAX_QUERIES_PER_HOUR`) that were set to zero, which meant that these were not distributed among SQL nodes. (Bug #30600321)

- Two buffers used for logging in `QMGR` were of insufficient size. (Bug #30598737)

  References: See also: Bug #30593511.

- Removed extraneous debugging output relating to `SPJ` from the node out logs. (Bug #30572315)

- When performing an initial restart of an NDB Cluster, each MySQL Server attached to it as an SQL node recognizes the restart, reinstalls the `ndb_schema` table from the data dictionary, and then clears all NDB schema definitions created prior to the restart. Because the data dictionary was cleared only after `ndb_schema` is reinstalled, installation sometimes failed due to `ndb_schema` having the same table ID as one of the tables from before the restart was performed. This issue is fixed by ensuring that the data dictionary is cleared before the `ndb_schema` table is reinstalled. (Bug #30488610)

- NDB sometimes made the assumption that the list of nodes containing index statistics was ordered, but this list is not always ordered in the same way on all nodes. This meant that in some cases NDB ignored a request to update index statistics, which could result in stale data in the index statistics tables. (Bug #30444982)
- When the optimizer decides to presort a table into a temporary table, before later tables are joined, the table to be sorted should not be part of a pushed join. Although logic was present in the abstract query plan interface to detect such query plans, that this did not detect correctly all situations using `filesort into temporary table`. This is changed to check whether a filesort descriptor has been set up; if so, the table content is sorted into a temporary file as its first step of accessing the table, which greatly simplifies interpretation of the structure of the join. We now also detect when the table to be sorted is a part of a pushed join, which should prevent future regressions in this interface. (Bug #30338585)
- When a node ID allocation request failed with **`NotMaster`** temporary errors, the node ID allocation was always retried immediately, without regard to the cause of the error. This caused a very high rate of retries, whose effects could be observed as an excessive number of `Alloc node id for node nnn failed` log messages (on the order of 15,000 messages per second). (Bug #30293495)
- For NDB tables having no explicit primary key, `NdbReceiverBuffer` could be allocated with too small a size. This was due to the fact that the attribute bitmap sent to NDB from the data nodes always includes the primary key. The extra space required for hidden primary keys is now taken into consideration in such cases. (Bug #30183466)
- When translating an NDB table created using `.frm` files in a previous version of NDB Cluster and storing it as a table object in the MySQL data dictionary, it was possible for the table object to be committed even when a mismatch had been detected between the table indexes in the MySQL data dictionary and those for the same table's representation the NDB dictionary. This issue did not occur for tables created in NDB 8.0, where it is not necessary to upgrade the table metadata in this fashion.

  This problem is fixed by making sure that all such comparisons are actually performed before the table object is committed, regardless of whether the originating table was created with or without the use of `.frm` files to store its metadata. (Bug #29783638)
- An error raised when obtaining cluster metadata caused a memory leak. (Bug #97737, Bug #30575163)

Changes in MySQL NDB Cluster 8.0.19 (2020-01-13, General Availability)

MySQL NDB Cluster 8.0.19 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.19 (see Changes in MySQL 8.0.19 (2020-01-13, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change:** The default value for the `ndb_autoincrement_prefetch_sz` server system variable has been increased to 512. (Bug #30316314)

- **Important Change:** NDB now supports more than 2 fragment replicas (up to a maximum of 4). Setting NoOfReplicas=3 or NoOfReplicas=4 is now fully covered in our internal testing and thus supported for use in production. (Bug #97479, Bug #97579, Bug #25261716, Bug #30501414, Bug #30528105, WL #8426)

- **Important Change:** Added the TransactionMemory data node configuration parameter which simplifies configuration of data node memory allocation for transaction operations. This is part of ongoing work on pooling of transactional and Local Data Manager (LDM) memory.

  The following parameters are incompatible with TransactionMemory and cannot be set in the config.ini configuration file if this parameter has been set:

  - MaxNoOfConcurrentIndexOperations
  - MaxNoOfFiredTriggers
  - MaxNoOfLocalOperations
  - MaxNoOfLocalScans

  If you attempt to set any of these incompatible parameters concurrently with TransactionMemory, the cluster management server cannot start.

  For more information, see the description of the TransactionMemory parameter and Parameters incompatible with TransactionMemory. See also Data Node Memory Management, for information about how memory resources are allocated by NDB Cluster data nodes. (Bug #96995, Bug #30344471, WL #12687)

- **Important Change:** The maximum or default values for several NDB Cluster data node configuration parameters have been changed in this release. These changes are listed here:

  - The maximum value for DataMemory is increased from 1 terabyte to 16 TB.
  - The maximum value for DiskPageBufferMemory is also increased from 1 TB to 16 TB.
  - The default value for StringMemory is decreased to 5 percent. Previously, this was 25 percent.
  - The default value for LcpScanProgressTimeout is increased from 60 seconds to 180 seconds.

  (WL #13382)

- **Performance:** Read from any fragment replica, which greatly improves the performance of table reads at a very low cost to table write performance, is now enabled by default for all NDB tables. This means both that the default value for the ndb_read_backup system variable is now ON, and that the value of the NDB_TABLE comment option READ_BACKUP is 1 when creating a new NDB table. (Previously, the default values were OFF and 0, respectively.)

  For more information, see Setting NDB Comment Options, as well as the description of the ndb_read_backup system variable. (WL #13383)

- **NDB Disk Data:** The latency of checkpoints for Disk Data files has been reduced when using non-volatile memory devices such as solid-state drives (especially those using NVMe for data transfer), separate physical drives for Disk Data files, or both. As part of this work, two new data node configuration parameters, listed here, have been introduced:

  - MaxDiskDataLatency sets a maximum on allowed latency for disk access, aborting transactions exceeding this amount of time to complete

- `DiskDataUsingSameDisk` makes it possible to take advantage of keeping Disk Data files on separate disks by increasing the rate at which Disk Data checkpoints can be made

This release also adds three new tables to the `ndbinfo` database. These tables, listed here, can assist with performance monitoring of Disk Data checkpointing:

- `diskstat` provides information about Disk Data tablespace reads, writes, and page requests during the previous 1 second
- `diskstats_1sec` provides information similar to that given by the `diskstat` table, but does so for each of the last 20 seconds
- `pgman_time_track_stats` table reports on the latency of disk operations affecting Disk Data tablespaces

For additional information, see Disk Data latency parameters. (WL #12924)

- Added the `ndb_metadata_sync` server system variable, which simplifies knowing when metadata synchronization has completed successfully. Setting this variable to `true` triggers immediate synchronization of all changes between the `NDB` dictionary and the MySQL data dictionary without regard to any values set for `ndb_metadata_check` or `ndb_metadata_check_interval`. When synchronization has completed, its value is automatically reset to `false`. (Bug #30406657)
- Added the `DedicatedNode` parameter for data nodes, API nodes, and management nodes. When set to true, this parameter prevents the management server from handing out this node's node ID to any node that does not request it specifically. Intended primarily for testing, this parameter may be useful in cases in which multiple management servers are running on the same host, and using the host name alone is not sufficient for distinguishing among processes of the same type. (Bug #91406, Bug #28239197)
- A stack trace is now written to the data node log on abnormal termination of a data node. (WL #13166)
- Automatic synchronization of metadata from the MySQL data dictionary to `NDB` now includes databases containing `NDB` tables. With this enhancement, if a table exists in `NDB`, and the table and the database it belongs to do not exist on a given SQL node, it is no longer necessary to create the database manually. Instead, the database, along with all `NDB` tables belonging to this database, should be created on the SQL node automatically. (WL #13490)

Bugs Fixed

- **Incompatible Change:** `ndb_restore` no longer restores shared users and grants to the `mysql.ndb_sql_metadata` table by default. A new command-line option `--include-stored-grants` is added to override this behavior and enable restoring of shared user and grant data and metadata.

  As part of this fix, `ndb_restore` can now also correctly handle an ordered index on a system table. (Bug #30237657)

  References: See also: Bug #29534239, Bug #30459246.

- **Incompatible Change:** The minimum value for the `RedoOverCommitCounter` data node configuration parameter has been increased from 0 to 1. The minimum value for the `RedoOverCommitLimit` data node configuration parameter has also been increased from 0 to 1.

  You should check the cluster global configuration file and make any necessary adjustments to values set for these parameters before upgrading. (Bug #29752703)

- **macOS:** On macOS, SQL nodes sometimes shut down unexpectedly during the binary log setup phase when starting the cluster. This occurred when there existed schemas whose names used

uppercase letters and `lower_case_table_names` was set to 2. This caused acquisition of metadata locks to be attempted using keys having the incorrect lettercase, and, subsequently, these locks to fail. (Bug #30192373)

- **Microsoft Windows; NDB Disk Data:** On Windows, restarting a data node other than the master when using Disk Data tables led to a failure in `TSMAN`. (Bug #97436, Bug #30484272)
- **Solaris:** When debugging, `ndbmtd` consumed all available swap space on Solaris 11.4 SRU 12 and later. (Bug #30446577)
- **Solaris:** The byte order used for numeric values stored in the `mysql.ndb_sql_metadata` table was incorrect on Solaris/Sparc. This could be seen when using `ndb_select_all` or `ndb_restore --print`. (Bug #30265016)
- **NDB Disk Data:** After dropping a disk data table on one SQL node, trying to execute a query against `INFORMATION_SCHEMA.FILES` on a different SQL node stalled at `Waiting for tablespace metadata lock`. (Bug #30152258)

  References: See also: Bug #29871406.
- **NDB Disk Data:** `ALTER TABLESPACE ... ADD DATAFILE` could sometimes hang while trying to acquire a metadata lock. (Bug #29871406)
- **NDB Disk Data:** Compatibility code for the Version 1 disk format used prior to the introduction of the Version 2 format in NDB 7.6 turned out not to be necessary, and is no longer used.
- Work done in NDB 8.0.18 to allow more nodes introduced long signal variants of several signals taking a bitmask as one of their arguments, and we started using these new long signal variants even if the previous (still supported) short variants would have been sufficient. This introduced several new opportunities for hitting `out of LongMessageBuffer` errors.

  To avoid this, now in such cases we use the short signal variants wherever possible. Some of the signals affected include `CM_REGCONF`, `CM_REGREF`, `FAIL_REP`, `NODE_FAILREP`, `ISOLATE_ORD`, `COPY_GCIREQ`, `START_RECREQ`, `NDB_STARTCONF`, and `START_LCP_REQ`. (Bug #30708009)

  References: See also: Bug #30707970.
- The fix made in NDB 8.0.18 for an issue in which a transaction was committed prematurely aborted the transaction if the table definition had changed midway, but failed in testing to free memory allocated by `getExtraMetadata()`. Now this memory is properly freed before aborting the transaction. (Bug #30576983)

  References: This issue is a regression of: Bug #29911440.
- Excessive allocation of attribute buffer when initializing data in `DBTC` led to preallocation of api connection records failing due to unexpectedly running out of memory. (Bug #30570264)
- Improved error handling in the case where `NDB` attempted to update a local user having the `NDB_STORED_USER` privilege but which could not be found in the `ndb_sql_metadata` table. (Bug #30556487)
- Failure of a transaction during execution of an `ALTER TABLE ... ALGORITHM=COPY` statement following the rename of the new table to the name of the original table but before dropping the original table caused `mysqld` to exit prematurely. (Bug #30548209)
- Non-MSI builds on Windows using `-DWITH_NDBCLUSTER` did not succeed unless the WiX toolkit was installed. (Bug #30536837)
- The `allowed_values` output from `ndb_config --xml --configinfo` for the `Arbitration` data node configuration parameter in NDB 8.0.18 was not consistent with that obtained in previous releases. (Bug #30529220)

  References: See also: Bug #30505003.

- A faulty `ndbrequire()` introduced when implementing partial local checkpoints assumed that `m_participatingLQH` must be clear when receiving `START_LCP_REQ`, which is not necessarily true when a failure happens for the master after sending `START_LCP_REQ` and before handling any `START_LCP_CONF` signals. (Bug #30523457)
- A local checkpoint sometimes hung when the master node failed while sending an `LCP_COMPLETE_REP` signal and it was sent to some nodes, but not all of them. (Bug #30520818)
- Added the `DUMP 9988` and `DUMP 9989` commands. (Bug #30520103)
- The management server did not handle all cases of `NODE_FAILREP` correctly. (Bug #30520066)
- With `SharedGlobalMemory` set to 0, some resources did not meet required minimums. (Bug #30411835)
- Execution of `ndb_restore --rebuild-indexes` together with the `--rewrite-database` and `--exclude-missing-tables` options did not create indexes for any tables in the target database. (Bug #30411122)
- When writing the schema operation into the `ndb_schema` table failed, the states in the `NDB_SCHEMA` object were not cleared, which led to the SQL node shutting down when it tried to free the object. (Bug #30402362)

  References: See also: Bug #30371590.
- When synchronizing extent pages it was possible for the current local checkpoint (LCP) to stall indefinitely if a `CONTINUEB` signal for handling the LCP was still outstanding when receiving the `FSWRITECONF` signal for the last page written in the extent synchronization page. The LCP could also be restarted if another page was written from the data pages. It was also possible that this issue caused `PREP_LCP` pages to be written at times when they should not have been. (Bug #30397083)
- If a transaction was aborted while getting a page from the disk page buffer and the disk system was overloaded, the transaction hung indefinitely. This could also cause restarts to hang and node failure handling to fail. (Bug #30397083, Bug #30360681)

  References: See also: Bug #30152258.
- Data node failures with the error `Another node failed during system restart...` occurred during a partial restart. (Bug #30368622)
- Automatic synchronization could potentially trigger an increase in the number of locks being taken on a particular metadata object at a given time, such as when a synchronization attempt coincided with a DDL or DML statement involving the same metadata object; competing locks could lead to the NDB deadlock detection logic penalizing the user action rather than the background synchronization. We fix this by changing all exclusive metadata lock acquisition attempts during auto-synchronization so that they use a timeout of 0 (rather than the 10 seconds previously allowed), which avoids deadlock detection and gives priority to the user action. (Bug #30358470)
- If a `SYNC_EXTENT_PAGES_REQ` signal was received by `PGMAN` while dropping a log file group as part of a partial local checkpoint, and thus dropping the page locked by this block for processing next, the LCP terminated due to trying to access the page after it had already been dropped. (Bug #30305315)
- The wrong number of bytes was reported in the cluster log for a completed local checkpoint. (Bug #30274618)

  References: See also: Bug #29942998.
- Added the new `ndb_mgm` client debugging commands `DUMP 2356` and `DUMP 2357`. (Bug #30265415)
- Executing `ndb_drop_table` using the `--help` option caused this program to terminate prematurely, and without producing any help output. (Bug #30259264)

- A `mysqld` trying to connect to the cluster, and thus trying to acquire the global schema lock (GSL) during setup, ignored the setting for `ndb-wait-setup` and hung indefinitely when the GSL had already been acquired by another `mysqld`, such as when it was executing an `ALTER TABLE` statement. (Bug #30242141)

- When a table containing self-referential foreign key (in other words, a foreign key referencing another column of the same table) was altered using the `COPY` algorithm, the foreign key definition was removed. (Bug #30233405)

- In MySQL 8.0, names of foreign keys explicitly provided by user are generated automatically in the SQL layer and stored in the data dictionary. Such names are of the form `[table_name]_ibfk_[#]` which align with the names generated by the `InnoDB` storage engine in MySQL 5.7. NDB 8.0.18 introduced a change in behavior by `NDB` such that it also uses the generated names, but in some cases, such as when tables were renamed, `NDB` still generated and used its own format for such names internally rather than those generated by the SQL layer and stored in the data dictionary, which led to the following issues:

  - Discrepancies in `SHOW CREATE TABLE` output and the contents of `INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS`

  - Improper metadata locking for foreign keys

  - Confusing names for foreign keys in error messages

  Now `NDB` also renames the foreign keys in such cases, using the names provided by the MySQL server, to align fully with those used by `InnoDB`. (Bug #30210839)

  References: See also: Bug #96508, Bug #30171959.

- When a table referenced by a foreign key was renamed, participating SQL nodes did not properly update the foreign key definitions for the referencing table in their data dictionaries during schema distribution. (Bug #30191068)

- Data node handling of failures of other data nodes could sometimes not be synchronized properly, such that two or more data nodes could see different nodes as the master node. (Bug #30188414)

- Some scan operations failed due to the presence of an old assert in `DbtupBuffer.cpp` that checked whether API nodes were using a version of the software previous to NDB 6.4. This was no longer necessary or correct, and has been removed. (Bug #30188411)

- When executing a global schema lock (GSL), `NDB` used a single `Ndb_table_guard` object for successive retires when attempting to obtain a table object reference; it was not possible for this to succeed after failing on the first attempt, since `Ndb_table_guard` assumes that the underlying object pointer is determined once only—at initialisation—with the previously retrieved pointer being returned from a cached reference thereafter.

  This resulted in infinite waits to obtain the GSL, causing the binlog injector thread to hang so that `mysqld` considered all `NDB` tables to be read-only. To avoid this problem, `NDB` now uses a fresh instance of `Ndb_table_guard` for each such retry. (Bug #30120858)

  References: This issue is a regression of: Bug #30086352.

- When upgrading an SQL node to NDB 8.0 from a previous release series, the `.frm` file whose contents are read and then installed in the data dictionary does not contain any information about foreign keys. This meant that foreign key information was not installed in the SQL node's data dictionary. This is fixed by using the foreign key information available in the NDB data dictionary to update the local MySQL data dictionary during table metadata upgrade. (Bug #30071043)

- Restoring tables with the `--disable-indexes` option resulted in the wrong table definition being installed in the MySQL data dictionary. This is because the serialized dictionary information (SDI) packed into the NDB dictionary's table definition is used to create the table object; the SDI definition

is updated only when the DDL change is done through the MySQL server. Installation of the wrong table definition meant that the table could not be opened until the indexes were re-created in the NDB dictionary again using `--rebuild-indexes`.

This is fixed by extending auto-synchronization such that it compares the SDI to the NDB dictionary table information and fails in cases in which the column definitions do not match. Mismatches involving indexes only are treated as temporary errors, with the table in question being detected again during the next round of change detection. (Bug #30000202, Bug #30414514)

- Restoring tables for which `MAX_ROWS` was used to alter partitioning from a backup made from NDB 7.4 to a cluster running NDB 7.6 did not work correctly. This is fixed by ensuring that the upgrade code handling `PartitionBalance` supplies a valid table specification to the `NDB` dictionary. (Bug #29955656)

- The number of data bytes for the summary event written in the cluster log when a backup completed was truncated to 32 bits, so that there was a significant mismatch between the number of log records and the number of data records printed in the log for this event. (Bug #29942998)

- `mysqld` sometimes aborted during a long `ALTER TABLE` operation that timed out. (Bug #29894768)

  References: See also: Bug #29192097.

- When an SQL node connected to `NDB`, it did not know whether it had previously connected to that cluster, and thus could not determine whether its data dictionary information was merely out of date, or completely invalid. This issue is solved by implementing a unique schema version identifier (schema UUID) to the `ndb_schema` table in `NDB` as well as to the `ndb_schema` table object in the data dictionary. Now, whenever a `mysqld` connects to a cluster as an SQL node, it can compare the schema UUID stored in its data dictionary against that which is stored in the `ndb_schema` table, and so know whether it is connecting for the first time. If so, the SQL node removes any entries that may be in its data dictionary. (Bug #29894166)

  References: See also: Bug #27543602.

- Improved log messages generated by table discovery and table metadata upgrades. (Bug #29894127)

- Using 2 LDM threads on a 2-node cluster with 10 threads per node could result in a partition imbalance, such that one of the LDM threads on each node was the primary for zero fragments. Trying to restore a multi-threaded backup from this cluster failed because the datafile for one LDM contained only the 12-byte data file header, which `ndb_restore` was unable to read. The same problem could occur in other cases, such as when taking a backup immediately after adding an empty node online.

  It was found that this occurred when `ODirect` was enabled for an EOF backup data file write whose size was less than 512 bytes and the backup was in the `STOPPING` state. This normally occurs only for an aborted backup, but could also happen for a successful backup for which an LDM had no fragments. We fix the issue by introducing an additional check to ensure that writes are skipped only if the backup actually contains an error which should cause it to abort. (Bug #29892660)

  References: See also: Bug #30371389.

- For `NDB` tables, `ALTER TABLE ... ALTER INDEX` did not work with `ALGORITHM=INPLACE`. (Bug #29700197)

- `ndb_restore` failed in testing on 32-bit platforms. This issue is fixed by increasing the size of the thread stack used by this tool from 64 KB to 128 KB. (Bug #29699887)

  References: See also: Bug #30406046.

- An unplanned shutdown of the cluster occurred due to an error in `DBTUP` while deleting rows from a table following an online upgrade. (Bug #29616383)

- In some cases the `SignalSender` class, used as part of the implementation of `ndb_mgmd` and `ndbinfo`, buffered excessive numbers of unneeded `SUB_GCP_COMPLETE_REP` and `API_REGCONF` signals, leading to unnecessary consumption of memory. (Bug #29520353)

  References: See also: Bug #20075747, Bug #29474136.

- The setting for the `BackupLogBufferSize` configuration parameter was not honored. (Bug #29415012)

- When `mysqld` was run with the `--upgrade=FORCE` option, it reported the following issues:

  ```
  [Warning]  Table 'mysql.ndb_apply_status' requires repair.
  [ERROR]    Table 'mysql.ndb_apply_status' repair failed.
  ```

  This was because `--upgrade=FORCE` causes a bootstrap system thread to run `CHECK TABLE FOR UPGRADE`, but `ha_ndbcluster::open()` refused to open the table before schema synchronization had completed, which eventually led to the reported conditions. (Bug #29305977)

  References: See also: Bug #29205142.

- When using explicit SHM connections, with `ShmSize` set to a value larger than the system's available shared memory, `mysqld` hung indefinitely on startup and produced no useful error messages. (Bug #28875553)

- The maximum global checkpoint (GCP) commit lag and GCP save timeout are recalculated whenever a node shuts down, to take into account the change in number of data nodes. This could lead to the unintentional shutdown of a viable node when the threshold decreased below the previous value. (Bug #27664092)

  References: See also: Bug #26364729.

- A transaction which inserts a child row may run concurrently with a transaction which deletes the parent row for that child. One of the transactions should be aborted in this case, lest an orphaned child row result.

  Before committing an insert on a child row, a read of the parent row is triggered to confirm that the parent exists. Similarly, before committing a delete on a parent row, a read or scan is performed to confirm that no child rows exist. When insert and delete transactions were run concurrently, their prepare and commit operations could interact in such a way that both transactions committed. This occurred because the triggered reads were performed using `LM_CommittedRead` locks (see `NdbOperation::LockMode`), which are not strong enough to prevent such error scenarios.

  This problem is fixed by using the stronger `LM_SimpleRead` lock mode for both triggered reads. The use of `LM_SimpleRead` rather than `LM_CommittedRead` locks ensures that at least one transaction aborts in every possible scenario involving transactions which concurrently insert into child rows and delete from parent rows. (Bug #22180583)

- Concurrent `SELECT` and `ALTER TABLE` statements on the same SQL node could sometimes block one another while waiting for locks to be released. (Bug #17812505, Bug #30383887)

- Failure handling in schema synchronization involves pushing warnings and errors to the binary logging thread. Schema synchronization is also retried in case of certain failures which could lead to an accumulation of warnings in the thread. Now such warnings and errors are cleared following each attempt at schema synchronization. (Bug #2991036)

- An `INCL_NODECONF` signal from any local blocks should be ignored when a node has failed, except in order to reset `c_nodeStartSlave.nodeId`. (Bug #96550, Bug #30187779)

- When returning Error 1022, `NDB` did not print the name of the affected table. (Bug #74218, Bug #19763093)

  References: See also: Bug #29700174.

Changes in MySQL NDB Cluster 8.0.18 (2019-10-14, Release Candidate)

MySQL NDB Cluster 8.0.18 is a new development release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.18 (see Changes in MySQL 8.0.18 (2019-10-14, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change:** The 63-byte limit on NDB database and table names has been removed. These identifiers may now take up to 64 bytes, as when using other MySQL storage engines. For more information, see Previous NDB Cluster Issues Resolved in NDB Cluster 8.0. (Bug #44940, Bug #11753491, Bug #27447958)

- **Important Change:** Implemented the NDB_STORED_USER privilege, which enables sharing of users, roles, and privileges across all SQL nodes attached to a given NDB Cluster. This replaces the distributed grant tables mechanism from NDB 7.6 and earlier versions of NDB Cluster, which was removed in NDB 8.0.16 due to its incompatibility with changes made to the MySQL privilege system in MySQL 8.0.

  A user or role which has this privilege is propagated, along with its (other) privileges to a MySQL server (SQL node) as soon as it connects to the cluster. Changes made to the privileges of the user or role are synchronized immediately with all connected SQL nodes.

  NDB_STORED_USER can be granted to users and roles other than reserved accounts such as mysql.session@localhost or mysql.infoschema@localhost. A role can be shared, but assigning a shared role to a user does not cause this user to be shared; the NDB_STORED_USER privilege must be granted to the user explicitly in order for the user to be shared between NDB Cluster SQL nodes.

  The NDB_STORED_USER privilege is always global and must be granted using ON *.*. This privilege is recognized only if the MySQL server enables support for the NDBCLUSTER storage engine.

  For usage information, see the description of NDB_STORED_USER. Privilege Synchronization and NDB_STORED_USER, has additional information on how NDB_STORED_USER and privilege synchronization work. For information on how this change may affect upgrades to NDB 8.0 from previous versions, see Upgrading and Downgrading NDB Cluster. (WL #12637)

  References: See also: Bug #29862601, Bug #29996547.

- **Important Change:** The maximum row size for an NDB table is increased from 14000 to 30000 bytes.

  As before, only the first 264 bytes of a BLOB or TEXT column count towards this total.

  The maximum offset for a fixed-width column of an NDB table is 8188 bytes; this is also unchanged from previous NDB Cluster releases.

For more information, see Limits Associated with Database Objects in NDB Cluster. (WL #13079, WL #11160)

References: See also: Bug #29485977, Bug #29024275.

- **Important Change:** A new binary format has been implemented for the NDB management server's cached configuration file, which is intended to support much larger numbers of nodes in a cluster than previously. Prior to this release, the configuration file supported a maximum of 16381 sections; this number is increased to 4G.

  Upgrades to the new format should not require any manual intervention, as the management server (and other cluster nodes) can still read the old format. For downgrades from this release or a later one to NDB 8.0.17 or earlier, it is necessary to remove the binary configuration files prior to starting the old management server binary, or start it using the `--initial` option.

  For more information, see Upgrading and Downgrading NDB Cluster. (WL #12453)

- **Important Change:** The maximum number of data nodes supported in a single NDB cluster is raised in this release from 48 to 144. The range of supported data node IDs is increased in conjunction with this enhancement to 1-144, inclusive.

  In previous releases, recommended node IDs for management nodes were 49 and 50. These values are still supported, but, if used, limit the maximum number of data nodes to 142. For this reason, the recommended node ID values for management servers are now 145 and 146.

  The maximum total supported number of nodes of all types in a given cluster is 255. This total is unchanged from previous releases.

  For a cluster running more than 48 data nodes, it is not possible to downgrade directly to a previous release that supports only 48 data nodes. In such cases, it is necessary to reduce the number of data nodes to 48 or fewer, and to make sure that all data nodes use node IDs that are less than 49.

  This change also introduces a new version (v2) of the format used for the data node `sysfile`, which records information such as the last global checkpoint index, restart status, and node group membership of each node (see NDB Cluster Data Node File System Directory). (WL #12680, WL #12564, WL #12876)

- **NDB Cluster APIs:** An alternative constructor for `NdbInterpretedCode` is now provided, which accepts an `NdbRecord` in place of a `Table` object. (Bug #29852377)

- **NDB Cluster APIs:** `NdbScanFilter::cmp()` and the following `NdbInterpretedCode` comparison methods can be now used to compare table column values:

  - `branch_col_eq()`
  - `branch_col_ge()`
  - `branch_col_gt()`
  - `branch_col_le()`
  - `branch_col_lt()`
  - `branch_col_ne()`

  When using any of these methods, the table column values to be compared must be of exactly the same type, including with respect to length, precision, and scale. In addition, in all cases, `NULL` is always considered by these methods to be less than any other value. You should also be aware

that, when used to compare table column values, `NdbScanFilter::cmp()` does not support all possible values of `BinaryCondition`.

For more information, see the descriptions of the individual API methods. (WL #13120)

- **NDB Client Programs:** The dependency of the `ndb_delete_all` utility on the `NDBT` library has been removed. This library, used in `NDB` development for testing, is not required for normal use. The visible change for users is that `ndb_delete_all` no longer prints `NDBT_ProgramExit - status` following completion of its run. Applications that depend upon this behavior should be updated to reflect this change when upgrading to this release. (WL #13223)

- `ndb_restore` now reports the specific `NDB` error number and message when it is unable to load a table descriptor from a backup `.ctl` file. This can happen when attempting to restore a backup taken from a later version of the NDB Cluster software to a cluster running an earlier version—for example, when the backup includes a table using a character set which is unknown to the version of `ndb_restore` being used to restore it. (Bug #30184265)

- The output from `DUMP 1000` in the `ndb_mgm` client has been extended to provide information regarding total data page usage. (Bug #29841454)

  References: See also: Bug #29929996.

- NDB Cluster's condition pushdown functionality has been extended as follows:

  - Expressions using any previously allowed comparisons are now supported.

  - Comparisons between columns in the same table and of the same type are now supported. The columns must be of exactly the same type.

  Example: Suppose there are two tables `t1` and `t2` created as shown here:

  ```
  CREATE TABLE t1 (a INT, b INT, c CHAR(10), d CHAR(5)) ENGINE=NDB;
  CREATE TABLE t2 LIKE t1;
  ```

  The following joins can now be pushed down to the data nodes:

  ```
  SELECT * FROM t1 JOIN t2 ON t2.a < t1.a+10;
  SELECT * FROM t1 JOIN t2 ON t2.a = t1.a+t1.b;
  SELECT * FROM t1 JOIN t2 ON t2.a = t1.a+t1.b;
  SELECT * FROM t1 JOIN t2 ON t2.d = SUBSTRING(t1.c,1,5);
  SELECT * FROM t1 JOIN t2 ON t2.c = CONCAT('foo',t1.d,'ba');
  ```

  Supported comparisons are `<`, `<=`, `>`, `>=`, `=`, and `<>`. (Bug #29685643, WL #12956, WL #13121)

- NDB Cluster now uses *`table_name`*`_fk_`*`N`* as the naming pattern for internally generated foreign keys, which is similar to the *`table_name`*`_ibfk_`*`N`* pattern used by `InnoDB`. (Bug #96508, Bug #30171959)

  References: See also: Bug #30210839.

- Added the `ndb_schema_dist_lock_wait_timeout` system variable to control how long to wait for a schema lock to be released when trying to update the SQL node's local data dictionary for one or more tables currently in use from the `NDB` data dictionary's metadata. If this synchronization has not yet occurred by the end of this time, the SQL node returns a warning that schema distribution did not succeed; the next time that the table for which distribution failed is accessed, `NDB` tries once again to synchronize the table metadata. (WL #10164)

- `NDB` table objects submitted by the metadata change monitor thread are now automatically checked for any mismatches and synchronized by the `NDB` binary logging thread. The status variable `Ndb_metadata_synced_count` added in this release shows the number of objects synchronized automatically; it is possible to see which objects have been synchronized by checking the cluster log. In addition, the new status variable `Ndb_metadata_blacklist_size` indicates the number of objects for which synchronization has failed. (WL #11914)

References: See also: Bug #30000202.

- It is now possible to build NDB for 64-bit ARM CPUs from the NDB Cluster sources. Currently, we do not provide any precompiled binaries for this platform. (WL #12928)
- Start times for the ndb_mgmd management node daemon have been significantly improved as follows:
  - More efficient handling of properties from configuration data can decrease startup times for the management server by a factor of 6 or more as compared with previous versions.
  - Host names not present in the management server's hosts file no longer create a bottleneck during startup, making ndb_mgmd start times up to 20 times shorter where these are used.

  (WL #13143)
- Columns of NDB tables can now be renamed online, using ALGORITHM=INPLACE. (WL #11734)

  References: See also: Bug #28609968.

Bugs Fixed

- **Important Change:** Because the current implementation for node failure handling cannot guarantee that even a single transaction of size MaxNoOfConcurrentOperations is completed in each round, this parameter is once again used to set a global limit on the total number of concurrent operations in all transactions within a single transaction coordinator instance. (Bug #96617, Bug #30216204)
- **Partitioning; NDB Disk Data:** Creation of a partitioned disk data table was unsuccessful due to a missing metadata lock on the tablespace specified in the CREATE TABLE statement. (Bug #28876892)
- **NDB Disk Data:** Tablespaces and data files are not tightly coupled in NDB, in the sense that they are represented by independent NdbDictionary objects. Thus, when metadata is restored using the ndb_restore tool, there was no guarantee that the tablespace and its associated datafile objects were restored at the same time. This led to the possibility that the tablespace mismatch was detected and automatically synchronized to the data dictionary before the datafile was restored to NDB. This issue also applied to log file groups and undo files.

  To fix this problem, the metadata change monitor now submits tablespaces and logfile groups only if their corresponding datafiles and undofiles actually exist in NDB. (Bug #30090080)
- **NDB Disk Data:** When a data node failed following creation and population of an NDB table having columns on disk, but prior to execution of a local checkpoint, it was possible to lose row data from the tablespace. (Bug #29506869)
- **NDB Cluster APIs:** The NDB API examples ndbapi_array_simple.cpp (see NDB API Simple Array Example) and ndbapi_array_using_adapter.cpp (see NDB API Simple Array Example Using Adapter) made assignments directly to a std::vector array instead of using push_back() calls to do so. (Bug #28956047)
- **MySQL NDB ClusterJ:** If ClusterJ was deployed as a separate module of a multi-module web application, when the application tried to create a new instance of a domain object, the exception java.lang.IllegalArgumentException: non-public interface is not defined by the given loader was thrown. It was because ClusterJ always tries to create a proxy class from which the domain object can be instantiated, and the proxy class is an implementation of the domain interface and the protected DomainTypeHandlerImpl::Finalizable interface. The class loaders of these two interfaces were different in the case, as they belonged to different modules running on the web server, so that when ClusterJ tried to create the proxy class using the domain object interface's class loader, the above-mentioned exception was thrown. This fix makes

the `Finalization` interface public so that the class loader of the web application would be able to access it even if it belongs to a different module from that of the domain interface. (Bug #29895213)

- **MySQL NDB ClusterJ:** ClusterJ sometimes failed with a segmentation fault after reconnecting to an NDB Cluster. This was due to ClusterJ reusing old database metadata objects from the old connection. With the fix, those objects are discarded before a reconnection to the cluster. (Bug #29891983)
- Faulty calculation of microseconds caused the internal `ndb_milli_sleep()` function to sleep for too short a time. (Bug #30211922)
- Once a data node is started, 95% of its configured `DataMemory` should be available for normal data, with 5% to spare for use in critical situations. During the node startup process, all of its configured `DataMemory` is usable for data, in order to minimize the risk that restoring the node data fails due to running out of data memory due to some dynamic memory structure using more pages for the same data than when the node was stopped. For example, a hash table grows differently during a restart than it did previously, since the order of inserts to the table differs from the historical order.

  The issue raised in this bug report occurred when a check that the data memory used plus the spare data memory did not exceed the value set for `DataMemory` failed at the point where the spare memory was reserved. This happened as the state of the data node transitioned from starting to started, when reserving spare pages. After calculating the number of reserved pages to be used for spare memory, and then the number of shared pages (that is, pages from shared global memory) to be used for this, the number of reserved pages already allocated was not taken into consideration. (Bug #30205182)

  References: See also: Bug #29616383.
- Removed a memory leak found in the `ndb_import` utility. (Bug #30192989)
- It was not possible to use `ndb_restore` and a backup taken from an NDB 8.0 cluster to restore to a cluster running NDB 7.6. (Bug #30184658)

  References: See also: Bug #30221717.
- When starting, a data node's local sysfile was not updated between the first completed local checkpoint and start phase 50. (Bug #30086352)
- In the `BACKUP` block, the assumption was made that the first record in `c_backups` was the local checkpoint record, which is not always the case. Now `NDB` loops through the records in `c_backups` to find the (correct) LCP record instead. (Bug #30080194)
- During node takeover for the master it was possible to end in the state `LCP_STATUS_IDLE` while the remaining data nodes were reporting their state as `LCP_TAB_SAVED`. This led to failure of the node when attempting to handle reception of a `LCP_COMPLETE_REP` signal since this is not expected when idle. Now in such cases local checkpoint handling is done in a manner that ensures that this node finishes in the proper state (`LCP_TAB_SAVED`). (Bug #30032863)
- When a MySQL Server built with `NDBCLUSTER` support was run on Solaris/x86, it failed during schema distribution. The root cause of the problem was an issue with the Developer Studio compiler used to build binaries for this platform when optimization level `-xO2` was used. This issue is fixed by using optimization level `-xO1` instead for `NDBCLUSTER` built for Solaris/x86. (Bug #30031130)

  References: See also: Bug #28585914, Bug #30014295.
- `NDB` used `free()` directly to deallocate `ndb_mgm_configuration` objects instead of calling `ndb_mgm_destroy_configuration()`, which correctly uses `delete` for deallocation. (Bug #29998980)
- Default configuration sections did not have the configuration section types set when unpacked into memory, which caused a memory leak since this meant that the section destructor would not destroy the entries for these sections. (Bug #29965125)

- No error was propagated when NDB failed to discover a table due to the table format being old and no longer supported, which could cause the NDB handler to retry the discovery operation endlessly and thereby hang. (Bug #29949096, Bug #29934763)
- During upgrade of an NDB Cluster when half of the data nodes were running NDB 7.6 while the remainder were running NDB 8.0, attempting to shut down those nodes which were running NDB 7.6 led to failure of one node with the error CHECK FAILEDNODEPTR.P->DBLQHFAI. (Bug #29912988, Bug #30141203)
- Altering a table in the middle of an ongoing transaction caused a table discovery operation which led to the transaction being committed prematurely; in addition, no error was returned when performing further updates as part of the same transaction.

  Now in such cases, the table discovery operation fails, when a transaction is in progress. (Bug #29911440)
- When performing a local checkpoint (LCP), a table's schema version was intermittently read as 0, which caused NDB LCP handling to treat the table as though it were being dropped. This could effect rebuilding of indexes offline by ndb_restore while the table was in the TABLE_READ_ONLY state. Now the function reading the schema version (getCreateSchemaVersion()) no longer not changes it while the table is read-only. (Bug #29910397)
- When an error occurs on an SQL node during schema distribution, information about this was written in the error log, but no indication was provided by the mysql client that the DDL statement in question was unsuccessful. Now in such cases, one or more generic warnings are displayed by the client to indicate that a given schema distribution operation has not been successful, with further information available in the error log of the originating SQL node. (Bug #29889869)
- Errors and warnings pushed to the execution thread during metadata synchronization and metadata change detection were not properly logged and cleared. (Bug #29874313)
- Altering a normal column to a stored generated column was performed online even though this is not supported. (Bug #29862463)
- A pushed join with ORDER BY did not always return the rows of the result in the specified order. This could occur when the optimizer used an ordered index to provide the ordering and the index used a column from the table that served as the root of the pushed join. (Bug #29860378)
- A number of issues in the Backup block for local checkpoints (LCPs) were found and fixed, including the following:
  - Bytes written to LCP part files were not always included in the LCP byte count.
  - The maximum record size for the buffer used for all LCP part files was not updated in all cases in which the table maximum record size had changed.
  - LCP surfacing could occur for LCP scans at times other than when receiving SCAN_FRAGCONF signals.
  - It was possible in some cases for the table currently being scanned to be altered in the middle of a scan request, which behavior is not supported.

  (Bug #29843373)

  References: See also: Bug #29485977.
- The requestInfo fields for the long and short forms of the LQHKEYREQ signal had different definitions; bits used for the key length in the short version were reused for flags in the long version, since the key length is implicit in the section length of the long version of the signal but it was possible for long LQHKEYREQ signals to contain a keylength in these same bits, which could be misinterpreted by the receiving local query handler, potentially leading to errors. Checks have now been implemented to make sure that this no longer happens. (Bug #29820838)

- The list of dropped shares could hold only one dropped NDB_SHARE instance for each key, which prevented NDB_SHARE instances with same key from being dropped multiple times while handlers held references to those NDB_SHARE instances. This interfered with keeping track of the memory allocated and being able to release it if mysqld shut down without all handlers having released their references to the shares. To resolve this issue, the dropped share list has been changed to use a list type which allows more than one NDB_SHARE with the same key to exist at the same time. (Bug #29812659, Bug #29812613)

- Removed an ndb_restore compile-time dependency on table names that was defined by the ndbcluster plugin. (Bug #29801100)

- When creating a table in parallel on multiple SQL nodes, the result was a race condition between checking that the table existed and opening the table, which caused CREATE TABLE IF NOT EXISTS to fail with Error 1. This was the result of two issues, described with their fixes here:

  1. Opening a table whose NDB_SHARE did not exist returned the non-descriptive error message ERROR 1296 (HY000): Got error 1 'Unknown error code' from NDBCLUSTER. This is fixed with a warning describing the problem in more detail, along with a more sensible error code.

     It was possible to open a table before schema synchronization was completed. This is fixed with a warning better describing the problem, along with an error indicating that cluster is not yet ready.

  In addition, this fixes a related issue in which creating indexes sometimes also failed with Error 1. (Bug #29793534, Bug #29871321)

- Previously, for a pushed condition, every request sent to NDB for a given table caused the generation of a new instance of NdbInterpretedCode. When joining tables, generation of multiple requests for all tables following the first table in the query plan is very likely; if the pushed condition had no dependencies on prior tables in the query plan, identical instances of NdbInterpretedCode were generated for each request, at a significant cost in wasted CPU cycles. Now such pushed conditions are identified and the required NdbInterpretedCode object is generated only once, and reused for every request sent for this table without the need for generating new code each time.

  This change also makes it possible for Scan Filter too large errors to be detected and set during query optimization, which corrects cases where the query plan shown was inaccurate because the indicated push of a condition later had to be undone during the execution phase. (Bug #29704575)

- Some instances of NdbScanFilter used in pushdown conditions were not generated properly due to FLOAT values being represented internally as having zero length. This led to more than the expected number of rows being returned from NDB, as shown by the value of Ndb_api_read_row_count. While the condition was re-evaluated by mysqld when generation of scan filter failed, the end result was still correct in such cases, but any performance gain expected from pushing the condition was lost. (Bug #29699347)

- When creating a table, NDB did not always determine correctly whether it exceeded the maximum allowed record size. (Bug #29698277)

- NDB index statistics are calculated based on the topology of one fragment of an ordered index; the fragment chosen in any particular index is decided at index creation time, both when the index is originally created, and when a node or system restart has recreated the index locally. This calculation is based in part on the number of fragments in the index, which can change when a table is reorganized. This means that, the next time that the node is restarted, this node may choose a different fragment, so that no fragments, one fragment, or two fragments are used to generate index statistics, resulting in errors from ANALYZE TABLE.

  This issue is solved by modifying the online table reorganization to recalculate the chosen fragment immediately, so that all nodes are aligned before and after any subsequent restart. (Bug #29534647)

- As part of initializing schema distribution, each data node must maintain a subscriber bitmap providing information about the API nodes that are currently subscribed to this data node. Previously, the size of the bitmap was hard-coded to `MAX_NODES` (256), which meant that large amounts of memory might be allocated but never used when the cluster had significantly fewer nodes than this value. Now the size of the bitmap is determined by checking the maximum API node ID used in the cluster configuration file. (Bug #29270539)

- The removal of the `mysql_upgrade` utility and its replacement by `mysqld --initialize` means that the upgrade procedure is executed much earlier than previously, possibly before `NDB` is fully ready to handle queries. This caused migration of the MySQL privilege tables from `NDB` to `InnoDB` to fail. (Bug #29205142)

- During a restart when the data nodes had started but not yet elected a president, the management server received a `node ID already in use` error, which resulted in excessive retries and logging. This is fixed by introducing a new error 1705 `Not ready for connection allocation yet` for this case.

  During a restart when the data nodes had not yet completed node failure handling, a spurious `Failed to allocate nodeID` error was returned. This is fixed by adding a check to detect an incomplete node start and to return error 1703 `Node failure handling not completed` instead.

  As part of this fix, the frequency of retries has been reduced for `not ready to alloc nodeID` errors, an error insert has been added to simulate a slow restart for testing purposes, and log messages have been reworded to indicate that the relevant node ID allocation errors are minor and only temporary. (Bug #27484514)

- `NDB` on Windows and macOS platforms did not always treat table names using mixed case consistently with `lower_case_table_names` = 2. (Bug #27307793)

- The process of selecting the transaction coordinator checked for “live” data nodes but not necessarily for those that were actually available. (Bug #27160203)

- The automatic metadata synchronization mechanism requires the binary logging thread to acquire the global schema lock before an object can be safely synchronized. When another thread had acquired this lock at the same time, the binary logging thread waited for up to `TransactionDeadlockDetectionTimeout` milliseconds and then returned failure if it was unsuccessful in acquiring the lock, which was unnecessary and which negatively impacted performance.

  This has been fixed by ensuring that the binary logging thread acquires the global schema lock, or else returns with an error, immediately. As part of this work, a new `OperationOptions` flag `OO_NOWAIT` has also been implemented in the NDB API. (WL #29740946)

Changes in MySQL NDB Cluster 8.0.17 (2019-07-22, Release Candidate)

MySQL NDB Cluster 8.0.17 is a new development release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.17 (see Changes in MySQL 8.0.17 (2019-07-22, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Schema operation timeout detection has been moved from the schema distribution client to the schema distribution coordinator, which now checks ongoing schema operations for timeout at regular intervals, marks participants that have timed out, emits suitable warnings when a schema operation timeout occurs, and prints a list of any ongoing schema operations at regular intervals.

  As part of this work, a new option `--ndb-schema-dist-timeout` makes it possible to set the number of seconds for a given SQL node to wait until a schema operation is marked as having timed out. (Bug #29556148)

- Added the status variable `Ndb_trans_hint_count_session`, which shows the number of transactions started in the current session that used hints. Compare this with `Ndb_api_trans_start_count_session` to get the proportion of all `NDB` transactions in the current session that have been able to use hinting. (Bug #29127040)

- When the cluster is in single user mode, the output of the `ndb_mgm SHOW` command now indicates which API or SQL node has exclusive access while this mode is in effect. (Bug #16275500)

Bugs Fixed

- **Important Change:** Attempting to drop, using the `mysql` client, an `NDB` table that existed in the MySQL data dictionary but not in `NDB` caused `mysqld` to fail with an error. This situation could occur when an `NDB` table was dropped using the `ndb_drop_table` tool or in an NDB API application using `dropTable()`. Now in such cases, `mysqld` drops the table from the MySQL data dictionary without raising an error. (Bug #29125206)

- **Important Change:** The dependency of `ndb_restore` on the `NDBT` library, which is used for internal testing only, has been removed. This means that the program no longer prints `NDBT_ProgramExit: ...` when terminating. Applications that depend upon this behavior should be updated to reflect this change when upgrading to this release. (WL #13117)

- **Packaging:** Added debug symbol packages to `NDB` distributions for `.deb`-based platforms which do not generate these automatically. (Bug #29040024)

- **NDB Disk Data:** If, for some reason, a disk data table exists in the NDB data dictionary but not in that of the MySQL server, the data dictionary is synchronized by installing the object. This can occur either during the schema synchronization phase when a MySQL server connects to an NDB Cluster, or during table discovery through a DML query or DDL statement.

  For disk data tables which used a tablespace for storage, the tablespace ID is stored as part of the data dictionary object, but this was not set during synchronization. (Bug #29597249)

- **NDB Disk Data:** Concurrent Disk Data table and tablespace DDL statements executed on the same SQL node caused a metadata lock deadlock. A DDL statement requires that an exclusive lock be taken on the object being modified and every such lock in turn requires that the global schema lock be acquired in `NDB`.

  To fix this issue, `NDB` now tracks when a global schema lock corresponding to an exclusive lock on a tablespace is taken. If a different global schema lock request fails while the first lock, NDB assumes that there is a deadlock. In this case, the deadlock is handled by having the new request release all locks it previously acquired, then retrying them at a later point. (Bug #29394407)

  References: See also: Bug #29175268.

- **NDB Disk Data:** Following execution of `ALTER TABLESPACE`, SQL statements on an existing table using the affected tablespace failed with error 3508 `Dictionary object id (id) does`

`not exist` where the object ID shown refers to the tablespace. Schema distribution of `ALTER TABLESPACE` involves dropping the old object from the data dictionary on a participating SQL node and creating a new one with a different dictionary object id, but the table object in the SQL node's data dictionary still used the old tablespace ID which rendered it unusable on the participants.

To correct this problem, tables using the tablespace are now retrieved and stored prior to the creation of the new tablespace, and then Updated the new object ID of the tablespace after it has been created in the data dictionary. (Bug #29389168)

- **NDB Replication:** Following the restart of an SQL node (`mysqld` process), the node's `ndb_apply_status` table was re-created as expected, but was never updated again afterwards. (Bug #30999967)
- **NDB Replication:** The `ndb_apply_status` table was created using the deprecated syntax `VARCHAR(255) BINARY`. `VARBINARY(255)` is now used instead for creating this table. (Bug #29807585)
- **NDB Replication:** Errors raised from replication settings by a `CREATE TABLE` statement were not properly checked, leading the user to believe (incorrectly) that the table was valid for this purpose. (Bug #29697052)
- **NDB Replication:** `NDB` did not handle binary logging of virtual generated columns of type `BLOB` correctly. Now such columns are always regarded as having zero length.
- **NDB Cluster APIs:** The memcached sources included with the NDB distribution would not build with `-Werror=format-security`. Now warnings are no longer treated as errors when compiling these files. (Bug #29512411)
- **NDB Cluster APIs:** It was not possible to scan a table whose `SingleUserMode` property had been set to `SingleUserModeReadWrite` or `SingleUserModeReadOnly`. (Bug #29493714)
- **NDB Cluster APIs:** The MGM API `ndb_logevent_get_next2()` function did not behave correctly on Windows and 32-bit Linux platforms. (Bug #94917, Bug #29609070)
- The version of Python expected by `ndb_setup.py` was not specified clearly on some platforms. (Bug #29818645)
- Lack of `SharedGlobalMemory` was incorrectly reported as lack of undo buffer memory, even though the cluster used no disk data tables. (Bug #29806771)

  References: This issue is a regression of: Bug #92125, Bug #28537319.
- Long `TCKEYREQ` signals did not always use the expected format when invoked from `TCINDXREQ` processing. (Bug #29772731)
- It was possible for an internal `NDB_SCHEMA_OBJECT` to be released too early or not at all; in addition, it was possible to create such an object that reused an existing key. (Bug #29759063)
- `ndb_restore` sometimes used `exit()` rather than `exitHandler()` to terminate the program, which could lead to resources not being properly freed. (Bug #29744353)
- Improved error message printed when the maximum offset for a `FIXED` column is exceeded. (Bug #29714670)
- Communication between the schema distribution client and the schema distribution coordinator is done using `NDB_SCHEMA_OBJECT` as well as by writing rows to the `ndb_schema` table in `NDB`. This allowed for the possibility of a number of different race conditions between when the registration of the schema operation and when the coordinator was notified of it.

  This fix addresses the following issues related to the situation just described:

  - The coordinator failed to abort active schema operations when the binary logging thread was restarted.

- Schema operations already registered were not aborted properly.
- The distribution client failed to detect correctly when schema distribution was not ready.
- The distribution client, when killed, exited without marking the current schema operation as failed.
- An operation in NDB_SHARE could be accessed without the proper locks being in place.

In addition, usage of the ndb_schema_share global pointer was removed, and replaced with detecting whether the schema distribution is ready by checking whether an operation for mysql.ndb_schema has been created in NDB_SHARE. (Bug #29639381)

- With DataMemory set to 200 GB, ndbmtd failed to start. (Bug #29630367)
- When a backup fails due to ABORT_BACKUP_ORD being received while waiting for buffer space, the backup calls closeScan() and then sends a SCAN_FRAGREQ signal to the DBLQH block to close the scan. As part of receiving SCAN_FRAGCONF in response, scanConf() is called on the operation object for the file record which in turn calls updateWritePtr() on the file system buffer (FsBuffer). At this point the length sent by updateWritePtr() should be 0, but in this case was not, which meant that the buffer did not have enough space even though it did not, the problem being that the size is calculated as scanStop - scanStart and these values were held over since the previous SCAN_FRAGCONF was received, and were not reset due to being out of buffer space.

  To avoid this problem, we now set scanStart = scanStop in confirmBufferData() (formerly scanConfExtra()) which is called as part of processing the SCAN_FRAGCONF, indirectly by scanConf() for the backup and first local checkpoint files, and directly for the LCP files which use only the operation record for the data buffer. (Bug #29601253)
- The setting for MaxDMLOperationsPerTransaction was not validated in a timely fashion, leading to data node failure rather than a management server error in the event that its value exceeded that of MaxNoOfConcurrentOperations. (Bug #29549572)
- Data nodes could fail due to an assert in the DBTC block under certain circumstances in resource-constrained environments. (Bug #29528188)
- An upgrade to NDB 7.6.9 or later from an earlier version could not be completed successfully if the redo log was filled to more than 25% of capacity. (Bug #29506844)
- When the DBSPJ block called the internal function lookup_resume() to schedule a previously enqueued operation, it used a correlation ID which could have been produced from its immediate ancestor in the execution order, and not its parent in the query tree as assumed. This could happen during execution of a SELECT STRAIGHT_JOIN query.

  Now NDB checks whether the execution ancestor is different from the query tree parent, and if not, performs a lookup of the query tree parent, and the parent's correlation ID is enqueued to be executed later. (Bug #29501263)
- When a new master took over, sending a MASTER_LCP_REQ signal and executing MASTER_LCPCONF from participating nodes, it expected that they had not completed the current local checkpoint under the previous master, which need not be true. (Bug #29487340, Bug #29601546)
- When restoring TINYBLOB columns, ndb_restore now treats them as having the BINARY character set. (Bug #29486538)
- When selecting a sorted result set from a query that included a LIMIT clause on a single table, and where the sort was executed as Using filesort and the ref access method was used on an ordered index, it was possible for the result set to be missing one or more rows. (Bug #29474188)
- Restoration of epochs by ndb_restore failed due to temporary redo errors. Now ndb_restore retries epoch updates when such errors occur. (Bug #29466089)

- `ndb_restore` tried to extract an 8-character substring of a table name when checking to determine whether or not the table was a blob table, regardless of the length of the name. (Bug #29465794)
- When a pushed join was used in combination with the `eq_ref` access method it was possible to obtain an incorrect join result due to the 1 row cache mechanism implemented in NDB 8.0.16 as part of the work done in that version to extend `NDB` condition pushdown by allowing referring values from previous tables. This issue is now fixed by turning off this caching mechanism and reading the row directly from the handler instead, when there is a pushed condition defined on the table. (Bug #29460314)
- Improved and made more efficient the conversion of rows by the `ha_ndbcluster` handler from the format used internally by `NDB` to that used by the MySQL server for columns that contain neither `BLOB` nor `BIT` values, which is the most common case. (Bug #29435461)
- A failed `DROP TABLE` could be attempted an infinite number of times in the event of a temporary error. Now in such cases, the number of retries is limited to 100. (Bug #29355155)
- `ndb_restore --restore-epoch` incorrectly reported the stop GCP as 1 less than the actual position. (Bug #29343655)
- A `SavedEvent` object in the `CMVMI` kernel block is written into a circular buffer. Such an object is split in two when wrapping at the end of the buffer; `NDB` looked beyond the end of the buffer instead of in the wrapped data at the buffer's beginning. (Bug #29336793)
- `NDB` did not compile with `-DWITH_SYSTEM_LIBS=ON` due to an incorrectly configured dependency on `zlib`. (Bug #29304517)
- Removed a memory leak found when running `ndb_mgmd --config-file` after compiling `NDB` with Clang 7. (Bug #29284643)
- Removed `clang` compiler warnings caused by usage of extra ; characters outside functions; these are incompatible with C++98. (Bug #29227925)
- Adding a column defined as `TIMESTAMP DEFAULT CURRENT_TIMESTAMP` to an `NDB` table is not supported with `ALGORITHM=INPLACE`. Attempting to do so now causes an error. (Bug #28128849)
- Added support which was missing in `ndb_restore` for conversions between the following sets of types:
  - `BLOB` and `BINARY` or `VARBINARY` columns
  - `TEXT` and `BLOB` columns
  - `BLOB` columns with unequal lengths
  - `BINARY` and `VARBINARY` columns with unequal lengths

  (Bug #28074988)
- Restore points in backups created with the `SNAPSHOTSTART` option (see Using The NDB Cluster Management Client to Create a Backup) were not always consistent with epoch boundaries. (Bug #27566346)

  References: See also: Bug #27497461.
- Neither the `MAX_EXECUTION_TIME` optimizer hint nor the `max_execution_time` system variable was respected for DDL statements or queries against `INFORMATION_SCHEMA` tables while an `NDB` global schema lock was in effect. (Bug #27538139)
- DDL operations were not always performed correctly on database objects including databases and tables, when multi-byte character sets were used for the names of either or both of these. (Bug #27150334)

- `ndb_import` did not always free up all resources used before exiting. (Bug #27130143)
- `NDBCLUSTER` subscription log printouts provided only 2 words of the bitmap (in most cases containing 8 words), which made it difficult to diagnose schema distribution issues. (Bug #22180480)
- For certain tables with very large rows and a very large primary key, `START BACKUP SNAPSHOTEND` while performing inserts into one of these tables or `START BACKUP SNAPSHOTSTART` with concurrent deletes could lead to data node errors.

  As part of this fix, `ndb_print_backup_file` can now read backup files created in very old versions of NDB Cluster (6.3 and earlier); in addition, this utility can now also read undo log files. (Bug #94654, Bug #29485977)
- When one of multiple SQL nodes which were connected to the cluster was down and then rejoined the cluster, or a new SQL node joined the cluster, this node did not use the data dictionary correctly, and thus did not always add, alter, or drop databases properly when synchronizing with the existing SQL nodes.

  Now, during schema distribution at startup, the SQL node compares all databases on the data nodes with those in its own data dictionary. If any database on the data nodes is found to be missing from the SQL node's data dictionary, the SQL Node installs it locally using `CREATE DATABASE`; the database is created using the default MySQL Server database properties currently in effect on this SQL node. (WL #12731)

Changes in MySQL NDB Cluster 8.0.16 (2019-04-25, Development Milestone)

MySQL NDB Cluster 8.0.16 is a new development release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.16 (see Changes in MySQL 8.0.16 (2019-04-25, General Availability)).

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

Deprecation and Removal Notes

- **Incompatible Change:** Distribution of privileges amongst MySQL servers connected to NDB Cluster, as implemented in NDB 7.6 and earlier, does not function in NDB 8.0, and most code supporting these has now been removed. When a `mysqld` detects such tables in `NDB`, it creates shadow tables local to itself using the `InnoDB` storage engine; these shadow tables are created on each MySQL server connected to an NDB cluster. Privilege tables using the `NDB` storage engine are not employed for access control; once all connected MySQL servers are upgraded, the privilege tables in `NDB` can be removed safely using `ndb_drop_table`.

  For compatibility reasons, `ndb_restore --restore-privilege-tables` can still be used to restore distributed privilege tables present in a backup taken from a previous release of NDB Cluster to a cluster running NDB 8.0. These tables are handled as described in the preceeding paragraph.

For additional information regarding upgrades from previous NDB Cluster release series to NDB 8.0, see Upgrading and Downgrading NDB Cluster. (WL #12507, WL #12511)

SQL Syntax Notes

- **Incompatible Change:** For consistency with `InnoDB`, the `NDB` storage engine now uses a generated constraint name if the `CONSTRAINT symbol` clause is not specified, or the `CONSTRAINT` keyword is specified without a `symbol`. In previous `NDB` releases, `NDB` used the `FOREIGN KEY index_name` value.

  This change described above may introduce incompatibilities for applications that depend on the previous foreign key constraint naming behavior. (Bug #29173134)

Functionality Added or Changed

- **Packaging:** A Docker image for this release can be obtained from https://hub.docker.com/r/mysql/mysql-cluster/. (Bug #96084, Bug #30010921)

- Allocation of resources in the transaction corrdinator (see The DBTC Block) is now performed using dynamic memory pools. This means that resource allocation determined by data node configuration parameters such as those discussed in Transaction parameters and Transaction temporary storage is now limited so as not to exceed the total resources available to the transaction coordinator.

  As part of this work, several new data node parameters controlling transactional resources in `DBTC`, listed here, have also been added. For more information about these new parameters, see Transaction resource allocation parameters. (Bug #29164271, Bug #29194843, WL #9756, WL #12523)

  References: See also: Bug #29131828.

- `NDB` backups can now be performed in a parallel fashion on individual data nodes using multiple local data managers (LDMs). (Previously, backups were done in parallel across data nodes, but were always serial within data node processes.) No special syntax is required for the `START BACKUP` command in the `ndb_mgm` client to enable this feature, but all data nodes must be using multiple LDMs. This means that data nodes must be running `ndbmtd` and they must be configured to use multiple LDMs prior to taking the backup (see Multi-Threading Configuration Parameters (ndbmtd)).

  The `EnableMultithreadedBackup` data node parameter introduced in this release is enabled (set to 1) by default. You can disable multi-threaded backups and force the creation of single-threaded backups by setting this parameter to 0 on all data nodes or in the `[ndbd default]` section of the cluster's global configuration file (`config.ini`).

  `ndb_restore` also now detects a multi-threaded backup and automatically attempts to restore it in parallel. It is also possible to restore backups taken in parallel to a previous version of NDB Cluster by slightly modifying the usual restore procedure.

  For more information about taking and restoring NDB Cluster backups that were created using parallelism on the data nodes, see Taking an NDB Backup with Parallel Data Nodes, and Restoring from a backup taken in parallel. (Bug #28563639, Bug #28993400, WL #8517)

- The `compile-cluster` script included in the `NDB` source distribution no longer supports in-source builds. (WL #12303)

- Building with `CMake3` is now supported by the `compile-cluster` script included in the `NDB` source distribution. (WL #12303)

- As part of its automatic synchronization mechanism, `NDB` now implements a metadata change monitor thread for detecting changes made to metadata for data objects such as tables, tablespaces, and log file groups with the MySQL data dictionary. This thread runs in the background, checking every 60 seconds for inconsistencies between the `NDB` dictionary and the MySQL data dictionary.

The monitor polling interval can be adjusted by setting the value of the `ndb_metadata_check_interval` system variable, and can be disabled altogether by setting `ndb_metadata_check` to OFF. The number of times that inconsistencies have been detected since `mysqld` was last started is shown as the status variable, `Ndb_metadata_detected_count`. (WL #11913)

- Condition pushdown is no longer limited to predicate terms referring to column values from the same table to which the condition was being pushed; column values from tables earlier in the query plan can now also be referred to from pushed conditions. This lets the data nodes filter out more rows (in parallel), leaving less work to be performed by a single `mysqld` process, which is expected to provide significant improvements in query performance.

  For more information, see Engine Condition Pushdown Optimization. (WL #12686)

Bugs Fixed

- **Important Change; NDB Disk Data:** `mysqldump` terminated unexpectedly when attempting to dump `NDB` disk data tables. The underlying reason for this was that `mysqldump` expected to find information relating to undo log buffers in the `EXTRA` column of the `INFORMATION_SCHEMA.FILES` table but this information had been removed in NDB 8.0.13. This information is now restored to the `EXTRA` column. (Bug #28800252)

- **Important Change:** When restoring to a cluster using data node IDs different from those in the original cluster, `ndb_restore` tried to open files corresponding to node ID 0. To keep this from happening, the `--nodeid` and `--backupid` options—neither of which has a default value—are both now explicitly required when invoking `ndb_restore`. (Bug #28813708)

- **Important Change:** Starting with this release, the default value of the `ndb_log_bin` system variable is now `FALSE`. (Bug #27135706)

- **Packaging; MySQL NDB ClusterJ:** `libndbclient` was missing from builds on some platforms. (Bug #28997603)

- **NDB Disk Data:** When a log file group had more than 18 undo logs, it was not possible to restart the cluster. (Bug #251155785)

  References: See also: Bug #28922609.

- **NDB Disk Data:** Concurrent `CREATE TABLE` statements using tablespaces caused deadlocks between metadata locks. This occurred when `Ndb_metadata_change_monitor` acquired exclusive metadata locks on tablespaces and logfile groups after detecting metadata changes, due to the fact that each exclusive metadata lock in turn acquired a global schema lock. This fix attempts to solve that issue by downgrading the locks taken by `Ndb_metadata_change_monitor` to `MDL_SHARED_READ`. (Bug #29175268)

  References: See also: Bug #29394407.

- **NDB Disk Data:** The error message returned when validation of `MaxNoOfOpenFiles` in relation to `InitialNoOfOpenFiles` failed has been improved to make the nature of the problem clearer to users. (Bug #28943749)

- **NDB Disk Data:** Schema distribution of `ALTER TABLESPACE` and `ALTER LOGFILE GROUP` statements failed on a participant MySQL server if the referenced tablespace or log file group did not exist in its data dictionary. Now in such cases, the effects of the statement are distributed successfully regardless of any initial mismatch between MySQL servers. (Bug #28866336)

- **NDB Disk Data:** Repeated execution of `ALTER TABLESPACE ... ADD DATAFILE` against the same tablespace caused data nodes to hang and left them, after being killed manually, unable to restart. (Bug #22605467)

- **NDB Replication:** A `DROP DATABASE` operation involving certain very large tables could lead to an unplanned shutdown of the cluster. (Bug #28855062)

- **NDB Replication:** When writes on the master—done in such a way that multiple changes affecting `BLOB` column values belonging to the same primary key were part of the same epoch—were replicated to the slave, Error 1022 occurred due to constraint violations in the `NDB$BLOB_id_part` table. (Bug #28746560)

- **NDB Cluster APIs:** `NDB` now identifies short-lived transactions not needing the reduction of lock contention provided by `NdbBlob::close()` and no longer invokes this method in cases (such as when autocommit is enabled) in which unlocking merely causes extra work and round trips to be performed prior to committing or aborting the transaction. (Bug #29305592)

  References: See also: Bug #49190, Bug #11757181.

- **NDB Cluster APIs:** When the most recently failed operation was released, the pointer to it held by `NdbTransaction` became invalid and when accessed led to failure of the NDB API application. (Bug #29275244)

- **NDB Cluster APIs:** When the `NDB` kernel's `SUMA` block sends a `TE_ALTER` event, it does not keep track of when all fragments of the event are sent. When `NDB` receives the event, it buffers the fragments, and processes the event when all fragments have arrived. An issue could possibly arise for very large table definitions, when the time between transmission and reception could span multiple epochs; during this time, `SUMA` could send a `SUB_GCP_COMPLETE_REP` signal to indicate that it has sent all data for an epoch, even though in this case that is not entirely true since there may be fragments of a `TE_ALTER` event still waiting on the data node to be sent. Reception of the `SUB_GCP_COMPLETE_REP` leads to closing the buffers for that epoch. Thus, when `TE_ALTER` finally arrives, NDB assumes that it is a duplicate from an earlier epoch, and silently discards it.

  We fix the problem by making sure that the `SUMA` kernel block never sends a `SUB_GCP_COMPLETE_REP` for any epoch in which there are unsent fragments for a `SUB_TABLE_DATA` signal.

  This issue could have an impact on NDB API applications making use of `TE_ALTER` events. (SQL nodes do not make any use of `TE_ALTER` events and so they and applications using them were not affected.) (Bug #28836474)

- When a pushed join executing in the `DBSPJ` block had to store correlation IDs during query execution, memory for these was allocated for the lifetime of the entire query execution, even though these specific correlation IDs are required only when producing the most recent batch in the result set. Subsequent batches require additional correlation IDs to be stored and allocated; thus, if the query took sufficiently long to complete, this led to exhaustion of query memory (error 20008). Now in such cases, memory is allocated only for the lifetime of the current result batch, and is freed and made available for re-use following completion of the batch. (Bug #29336777)

  References: See also: Bug #26995027.

- When comparing or hashing a fixed-length string that used a `NO_PAD` collation, any trailing padding characters (typically spaces) were sent to the hashing and comparison functions such that they became significant, even though they were not supposed to be. Now any such trailing spaces are trimmed from a fixed-length string whenever a `NO_PAD` collation is specified.

  Note

  Since `NO_PAD` collations were introduced as part of UCA-9.0 collations in MySQL 8.0, there should be no impact relating to this fix on upgrades to NDB 8.0 from previous GA releases of NDB Cluster.

  (Bug #29322313)

- When a `NOT IN` or `NOT BETWEEN` predicate was evaluated as a pushed condition, `NULL` values were not eliminated by the condition as specified in the SQL standard. (Bug #29232744)

  References: See also: Bug #28672214.

- Internally, `NDB` treats `NULL` as less than any other value, and predicates of the form *`column < value`* or *`column <= value`* are checked for possible nulls. Predicates of the form *`value > column`* or *`value >= column`* were not checked, which could lead to errors. Now in such cases, these predicates are rewritten so that the column comes first, so that they are also checked for the presence of `NULL`. (Bug #29231709)

  References: See also: Bug #92407, Bug #28643463.

- After folding of constants was implemented in the MySQL Optimizer, a condition containing a `DATE` or `DATETIME` literal could no longer be pushed down by `NDB`. (Bug #29161281)

- When a join condition made a comparison between a column of a temporal data type such as `DATE` or `DATETIME` and a constant of the same type, the predicate was pushed if the condition was expressed in the form *`column operator constant`*, but not when in inverted order (as *`constant inverse_operator column`*). (Bug #29058732)

- When processing a pushed condition, `NDB` did not detect errors or warnings thrown when a literal value being compared was outside the range of the data type it was being compared with,and thus truncated. This could lead to excess or missing rows in the result. (Bug #29054626)

- If an `EQ_REF` or `REF` key in the child of a pushed join referred to any columns of a table not a member of the pushed join, this table was not an `NDB` table (because its format was of nonnative endianness), and the data type of the column being joined on was stored in an endian-sensitive format, then the key generated was generated, likely resulting in the return of an (invalid) empty join result.

  Since only big endian platforms may store tables in nonnative (little endian) formats, this issue was expected only on such platforms, most notably SPARC, and not on x86 platforms. (Bug #29010641)

- API and data nodes running NDB 7.6 and later could not use an existing parsed configuration from an earlier release series due to being overly strict with regard to having values defined for configuration parameters new to the later release, which placed a restriction on possible upgrade paths. Now NDB 7.6 and later are less strict about having all new parameters specified explicitly in the configuration which they are served, and use hard-coded default values in such cases. (Bug #28993400)

- NDB 7.6 SQL nodes hung when trying to connect to an NDB 8.0 cluster. (Bug #28985685)

- The schema distribution data maintained in the `NDB` binary logging thread keeping track of the number of subscribers to the `NDB` schema table always allocated some memory structures for 256 data nodes regardless of the actual number of nodes. Now `NDB` allocates only as many of these structures as are actually needed. (Bug #28949523)

- Added `DUMP 406` (`NdbfsDumpRequests`) to provide `NDB` file system information to global checkpoint and local checkpoint stall reports in the node logs. (Bug #28922609)

- When a joined table was eliminated early as not pushable, it could not be referred to in any subsequent join conditions from other tables without eliminating those conditions from consideration even if those conditions were otherwise pushable. (Bug #28898811)

- When starting or restarting an SQL node and connecting to a cluster where `NDB` was already started, `NDB` reported Error 4009 `Cluster Failure` because it could not acquire a global schema lock. This was because the MySQL Server as part of initialization acquires exclusive metadata locks in order to modify internal data structures, and the `ndbcluster` plugin acquires the global schema lock. If the connection to `NDB` was not yet properly set up during `mysqld` initialization, `mysqld` received a warning from `ndbcluster` when the latter failed to acquire global schema lock, and printed it to the log file, causing an unexpected error in the log. This is fixed by not pushing any

warnings to background threads when failure to acquire a global schema lock occurs and pushing the NDB error as a warning instead. (Bug #28898544)

- A race condition between the DBACC and DBLQH kernel blocks occurred when different operations in a transaction on the same row were concurrently being prepared and aborted. This could result in DBTUP attempting to prepare an operation when a preceding operation had been aborted, which was unexpected and could thus lead to undefined behavior including potential data node failures. To solve this issue, DBACC and DBLQH now check that all dependencies are still valid before attempting to prepare an operation.

  Note

  This fix also supersedes a previous one made for a related issue which was originally reported as Bug #28500861.

  (Bug #28893633)

- Where a data node was restarted after a configuration change whose result was a decrease in the sum of MaxNoOfTables, MaxNoOfOrderedIndexes, and MaxNoOfUniqueHashIndexes, it sometimes failed with a misleading error message which suggested both a temporary error and a bug, neither of which was the case.

  The failure itself is expected, being due to the fact that there is at least one table object with an ID greater than the (new) sum of the parameters just mentioned, and that this table cannot be restored since the maximum value for the ID allowed is limited by that sum. The error message has been changed to reflect this, and now indicates that this is a permanent error due to a problem configuration. (Bug #28884880)

- The ndbinfo.cpustat table reported inaccurate information regarding send threads. (Bug #28884157)

- Execution of an LCP_COMPLETE_REP signal from the master while the LCP status was IDLE led to an assertion. (Bug #28871889)

- NDB now provides on-the-fly .frm file translation during discovery of tables created in versions of the software that did not support the MySQL Data Dictionary. Previously, such translation of tables that had old-style metadata was supported only during schema synchronization during MySQL server startup, but not subsequently, which led to errors when NDB tables having old-style metadata, created by ndb_restore and other such tools after mysqld had been started, were accessed using SHOW CREATE TABLE or SELECT; these tables were usable only after restarting mysqld. With this fix, the restart is no longer required. (Bug #28841009)

- An in-place upgrade to an NDB 8.0 release from an earlier relase did not remove .ndb files, even though these are no longer used in NDB 8.0. (Bug #28832816)

- Removed storage/ndb/demos and the demonstration scripts and support files it contained from the source tree. These were obsolete and unmaintained, and did not function with any current version of NDB Cluster.

  Also removed storage/ndb/include/newtonapi, which included files relating to an obsolete and unmaintained API not supported in any release of NDB Cluster, as well as references elsewhere to these files. (Bug #28808766)

- There was no version compatibility table for NDB 8.x; this meant that API nodes running NDB 8.0.13 or 7.6.x could not connect to data nodes running NDB 8.0.14. This issue manifested itself for NDB API users as a failure in wait_until_ready(). (Bug #28776365)

  References: See also: Bug #18886034, Bug #18874849.

- Issuing a STOP command in the ndb_mgm client caused ndbmtd processes which had recently been added to the cluster to hang in Phase 4 during shutdown. (Bug #28772867)

- A fix for a previous issue disabled the usage of pushed conditions for lookup type (`eq_ref`) operations in pushed joins. It was thought at the time that not pushing a lookup condition would not have any measurable impact on performance, since only a single row could be eliminated if the condition failed. The solution implemented at that time did not take into account the possibility that, in a pushed join, a lookup operation could be a parent operation for other lookups, and even scan operations, which meant that eliminating a single row could actually result in an entire branch being eliminated in error. (Bug #28728603)

  References: This issue is a regression of: Bug #27397802.

- When a local checkpoint (LCP) was complete on all data nodes except one, and this node failed, `NDB` did not continue with the steps required to finish the LCP. This led to the following issues:

  No new LCPs could be started.

  Redo and Undo logs were not trimmed and so grew excessively large, causing an increase in times for recovery from disk. This led to write service failure, which eventually led to cluster shutdown when the head of the redo log met the tail. This placed a limit on cluster uptime.

  Node restarts were no longer possible, due to the fact that a data node restart requires that the node's state be made durable on disk before it can provide redundancy when joining the cluster. For a cluster with two data nodes and two fragment replicas, this meant that a restart of the entire cluster (system restart) was required to fix the issue (this was not necessary for a cluster with two fragment replicas and four or more data nodes). (Bug #28728485, Bug #28698831)

  References: See also: Bug #11757421.

- The pushability of a condition to `NDB` was limited in that all predicates joined by a logical `AND` within a given condition had to be pushable to `NDB` in order for the entire condition to be pushed. In some cases this severely restricted the pushability of conditions. This fix breaks up the condition into its components, and evaluates the pushability of each predicate; if some of the predicates cannot be pushed, they are returned as a remainder condition which can be evaluated by the MySQL server. (Bug #28728007)

- Running `ANALYZE TABLE` on an `NDB` table with an index having longer than the supported maximum length caused data nodes to fail. (Bug #28714864)

- It was possible in certain cases for nodes to hang during an initial restart. (Bug #28698831)

  References: See also: Bug #27622643.

- When a condition was pushed to a storage engine, it was re-evaluated by the server, in spite of the fact that only rows matching the pushed condition should ever be returned to the server in such cases. (Bug #28672214)

- In some cases, one and sometimes more data nodes underwent an unplanned shutdown while running `ndb_restore`. This occurred most often, but was not always restircted to, when restoring to a cluster having a different number of data nodes from the cluster on which the original backup had been taken.

  The root cause of this issue was exhaustion of the pool of `SafeCounter` objects, used by the `DBDICT` kernel block as part of executing schema transactions, and taken from a per-block-instance pool shared with protocols used for `NDB` event setup and subscription processing. The concurrency of event setup and subscription processing is such that the `SafeCounter` pool can be exhausted; event and subscription processing can handle pool exhaustion, but schema transaction processing could not, which could result in the node shutdown experienced during restoration.

  This problem is solved by giving `DBDICT` schema transactions an isolated pool of reserved `SafeCounters` which cannot be exhausted by concurrent `NDB` event activity. (Bug #28595915)

- When a backup aborted due to buffer exhaustion, synchronization of the signal queues prior to the expected drop of triggers for insert, update, and delete operations resulted in abort signals being processed before the `STOP_BACKUP` phase could continue. The abort changed the backup status to `ABORT_BACKUP_ORD`, which led to an unplanned shutdown of the data node since resuming `STOP_BACKUP` requires that the state be `STOP_BACKUP_REQ`. Now the backup status is not set to `STOP_BACKUP_REQ` (requesting the backup to continue) until after signal queue synchronization is complete. (Bug #28563639)
- The output of `ndb_config --configinfo --xml --query-all` now shows that configuration changes for the `ThreadConfig` and `MaxNoOfExecutionThreads` data node parameters require system initial restarts (`restart="system" initial="true"`). (Bug #28494286)
- After a commit failed due to an error, `mysqld` shut down unexpectedly while trying to get the name of the table involved. This was due to an issue in the internal function `ndbcluster_print_error()`. (Bug #28435082)
- API nodes should observe that a node is moving through `SL_STOPPING` phases (graceful stop) and stop using the node for new transactions, which minimizes potential disruption in the later phases of the node shutdown process. API nodes were only informed of node state changes via periodic heartbeat signals, and so might not be able to avoid interacting with the node shutting down. This generated unnecessary failures when the heartbeat interval was long. Now when a data node is being gracefully stopped, all API nodes are notified directly, allowing them to experience minimal disruption. (Bug #28380808)
- `ndb_config --diff-default` failed when trying to read a parameter whose default value was the empty string (`""`). (Bug #27972537)
- `ndb_restore` did not restore autoincrement values correctly when one or more staging tables were in use. As part of this fix, we also in such cases block applying of the `SYSTAB_0` backup log, whose content continued to be applied directly based on the table ID, which could ovewrite the autoincrement values stored in `SYSTAB_0` for unrelated tables. (Bug #27917769, Bug #27831990)

  References: See also: Bug #27832033.
- `ndb_restore` employed a mechanism for restoring autoincrement values which was not atomic, and thus could yield incorrect autoincrement values being restored when multiple instances of `ndb_restore` were used in parallel. (Bug #27832033)

  References: See also: Bug #27917769, Bug #27831990.
- Executing `SELECT * FROM INFORMATION_SCHEMA.TABLES` caused SQL nodes to restart in some cases. (Bug #27613173)
- When tables with `BLOB` columns were dropped and then re-created with a different number of `BLOB` columns the event definitions for monitoring table changes could become inconsistent in certain error situations involving communication errors when the expected cleanup of the corresponding events was not performed. In particular, when the new versions of the tables had more `BLOB` columns than the original tables, some events could be missing. (Bug #27072756)
- When query memory was exhausted in the `DBSPJ` kernel block while storing correlation IDs for deferred operations, the query was aborted with error status 20000 `Query aborted due to out of query memory`. (Bug #26995027)

  References: See also: Bug #86537.
- When running a cluster with 4 or more data nodes under very high loads, data nodes could sometimes fail with Error 899 `Rowid already allocated`. (Bug #25960230)
- `mysqld` shut down unexpectedly when a purge of the binary log was requested before the server had completely started, and it was thus not yet ready to delete rows from the `ndb_binlog_index` table. Now when this occurs, requests for any needed purges of the `ndb_binlog_index` table are saved in a queue and held for execution when the server has completely started. (Bug #25817834)

- `MaxBufferedEpochs` is used on data nodes to avoid excessive buffering of row changes due to lagging `NDB` event API subscribers; when epoch acknowledgements from one or more subscribers lag by this number of epochs, an asynchronous disconnection is triggered, allowing the data node to release the buffer space used for subscriptions. Since this disconnection is asynchronous, it may be the case that it has not completed before additional new epochs are completed on the data node, resulting in new epochs not being able to seize GCP completion records, generating warnings such as those shown here:

  ```
  [ndbd] ERROR    -- c_gcp_list.seize() failed...

  ...

  [ndbd] WARNING  -- ACK wo/ gcp record...
  ```

  And leading to the following warning:

  ```
  Disconnecting node %u because it has exceeded MaxBufferedEpochs
  (100 > 100), epoch ....
  ```

  This fix performs the following modifications:

  - Modifies the size of the GCP completion record pool to ensure that there is always some extra headroom to account for the asynchronous nature of the disconnect processing previously described, thus avoiding `c_gcp_list` seize failures.
  - Modifies the wording of the `MaxBufferedEpochs` warning to avoid the contradictory phrase “100 > 100”.

  (Bug #20344149)

- Asynchronous disconnection of `mysqld` from the cluster caused any subsequent attempt to start an NDB API transaction to fail. If this occurred during a bulk delete operation, the SQL layer called `HA::end_bulk_delete()`, whose implementation by `ha_ndbcluster` assumed that a transaction had been started, and could fail if this was not the case. This problem is fixed by checking that the transaction pointer used by this method is set before referencing it. (Bug #20116393)

- Removed warnings raised when compiling `NDB` with Clang 6. (Bug #93634, Bug #29112560)

- When executing the redo log in debug mode it was possible for a data node to fail when deallocating a row. (Bug #93273, Bug #28955797)

- An `NDB` table having both a foreign key on another `NDB` table using `ON DELETE CASCADE` and one or more `TEXT` or `BLOB` columns leaked memory.

  As part of this fix, `ON DELETE CASCADE` is no longer supported for foreign keys on `NDB` tables when the child table contains a column that uses any of the `BLOB` or `TEXT` types. (Bug #89511, Bug #27484882)

Changes in MySQL NDB Cluster 8.0.15 (Not released)

MySQL NDB Cluster 8.0.15 was not released. NDB Cluster 8.0.14 is followed by NDB Cluster 8.0.16; users of NDB 8.0.14 should upgrade to 8.0.16 when the latter version becomes available.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

Changes in MySQL NDB Cluster 8.0.14 (2019-01-21, Development Milestone)

MySQL NDB Cluster 8.0.14 is a new development release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.14 (see Changes in MySQL 8.0.14 (2019-01-21, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Performance:** This release introduces a number of significant improvements in the performance of scans; these are listed here:
  - Row checksums help detect hardware issues, but do so at the expense of performance. NDB now offers the possibility of disabling these by setting the new `ndb_row_checksum` server system variable to 0; doing this means that row checksums are not used for new or altered tables. This can have a significant impact (5 to 10 percent, in some cases) on performance for all types of queries. This variable is set to 1 by default, to provide compatibility with the previous behavior.
  - A query consisting of a scan can execute for a longer time in the LDM threads when the queue is not busy.
  - Previously, columns were read before checking a pushed condition; now checking of a pushed condition is done before reading any columns.
  - Performance of pushed joins should see significant improvement when using range scans as part of join execution.

  (WL #11722)
- **NDB Disk Data:** NDB now implements schema distribution of disk data objects including tablespaces and log file groups by SQL nodes when they connect to a cluster, just as it does for NDB databases and in-memory tables. This eliminates a possible mismatch between the MySQL data dictionary and the NDB dictionary following a native backup and restore that could arise when disk data tablespaces and undo log file groups were restored to the NDB dictionary, but not to the MySQL Server's data dictionary. (WL #12172)
- **NDB Disk Data:** NDB now makes use of the MySQL data dictionary to ensure correct distribution of tablespaces and log file groups across all cluster SQL nodes when connecting to the cluster. (WL #12333)
- The extra metadata property for NDB tables is now used to store information from the MySQL data dictionary. Because this information is significantly larger than the binary representation previously stored here (a `.frm` file, no longer used), the hard-coded size limit for this extra metadata has been increased.

  This change can have an impact on downgrades: Trying to read NDB tables created in NDB 8.0.14 and later may cause data nodes running NDB 8.0.13 or earlier to fail on startup with NDB error code 2355 `Failure to restore schema: Permanent error, external action needed:`

`Resource configuration error.` This can happen if the table's metadata exceeds 6K in size, which was the old limit. Tables created in NDB 8.0.13 and earlier can be read by later versions without any issues.

For more information, see Changes in NDB table extra metadata, and See also MySQL Data Dictionary. (Bug #27230681, WL #10665)

Bugs Fixed

- **Packaging:** Expected NDB header files were in the `devel` RPM package instead of `libndbclient-devel`. (Bug #84580, Bug #26448330)
- **ndbmemcache:** `libndbclient.so` was not able to find and load `libssl.so`, which could cause issues with `ndbmemcache` and Java-based programs using `NDB`. (Bug #26824659)

  References: See also: Bug #27882088, Bug #28410275.
- **MySQL NDB ClusterJ:** The `ndb.clusterj` test for NDB 8.0.13 failed when being run more than once. This was deal to a new, stricter rule with NDB 8.0.13 that did not allow temporary files being left behind in the variable folder of `mysql-test-run (mtr)`. With this fix, the temporary files are deleted before the test is executed. (Bug #28279038)
- **MySQL NDB ClusterJ:** A `NullPointerException` was thrown when a full table scan was performed with ClusterJ on tables containing either a BLOB or a TEXT field. It was because the proper object initializations were omitted, and they have now been added by this fix. (Bug #28199372, Bug #91242)
- The `version_comment` system variable was not correctly configured in `mysqld` binaries and returned a generic pattern instead of the proper value. This affected all NDB Cluster binary releases with the exception of `.deb` packages. (Bug #29054235)
- Trying to build from source using `-DWITH_NDBCLUSTER` and `-Werror` failed with GCC 8. (Bug #28707282)
- When copying deleted rows from a live node to a node just starting, it is possible for one or more of these rows to have a global checkpoint index equal to zero. If this happened at the same time that a full local checkpoint was started due to the undo log getting full, the `LCP_SKIP` bit was set for a row having GCI = 0, leading to an unplanned shutdown of the data node. (Bug #28372628)
- `ndbmtd` sometimes experienced a hang when exiting due to log thread shutdown. (Bug #28027150)
- `NDB` has an upper limit of 128 characters for a fully qualified table name. Due to the fact that `mysqld` names `NDB` tables using the format *`database_name/catalog_name/table_name`*, where *`catalog_name`* is always `def`, it is possible for statements such as `CREATE TABLE` to fail in spite of the fact that neither the table name nor the database name exceeds the 63-character limit imposed by `NDB`. The error raised in such cases was misleading and has been replaced. (Bug #27769521)

  References: See also: Bug #27769801.
- When the `SUMA` kernel block receives a `SUB_STOP_REQ` signal, it executes the signal then replies with `SUB_STOP_CONF`. (After this response is relayed back to the API, the API is open to send more `SUB_STOP_REQ` signals.) After sending the `SUB_STOP_CONF`, SUMA drops the subscription if no subscribers are present, which involves sending multiple `DROP_TRIG_IMPL_REQ` messages to `DBTUP`. LocalProxy can handle up to 21 of these requests in parallel; any more than this are queued in the Short Time Queue. When execution of a `DROP_TRIG_IMPL_REQ` was delayed, there was a chance for the queue to become overloaded, leading to a data node shutdown with `Error in short time queue`.

  This issue is fixed by delaying the execution of the `SUB_STOP_REQ` signal if `DBTUP` is already handling `DROP_TRIG_IMPL_REQ` signals at full capacity, rather than queueing up the `DROP_TRIG_IMPL_REQ` signals. (Bug #26574003)

- `ndb_restore` returned -1 instead of the expected exit code in the event of an index rebuild failure. (Bug #25112726)
- When starting, a data node copies metadata, while a local checkpoint updates metadata. To avoid any conflict, any ongoing LCP activity is paused while metadata is being copied. An issue arose when a local checkpoint was paused on a given node, and another node that was also restarting checked for a complete LCP on this node; the check actually caused the LCP to be completed before copying of metadata was complete and so ended the pause prematurely. Now in such cases, the LCP completion check waits to complete a paused LCP until copying of metadata is finished and the pause ends as expected, within the LCP in which it began. (Bug #24827685)
- `ndbout` and `ndberr` became invalid after exiting from `mgmd_run()`, and redirecting to them before the next call to `mgmd_run()` caused a segmentation fault, during an `ndb_mgmd` service restart. This fix ensures that `ndbout` and `ndberr` remain valid at all times. (Bug #17732772, Bug #28536919)
- `NdbScanFilter` did not always handle `NULL` according to the SQL standard, which could result in sending non-qualifying rows to be filtered (otherwise not necessary) by the MySQL server. (Bug #92407, Bug #28643463)

  References: See also: Bug #93977, Bug #29231709.
- The internal function `ndb_my_error()` was used in `ndbcluster_get_tablespace_statistics()` and `prepare_inplace_alter_table()` to report errors when the function failed to interact with `NDB`. The function was expected to push the NDB error as warning on the stack and then set an error by translating the NDB error to a MySQL error and then finally call `my_error()` with the translated error. When calling `my_error()`, the function extracts a format string that may contain placeholders and use the format string in a function similar to `sprintf()`, which in this case could read arbitrary memory leading to a segmentation fault, due to the fact that `my_error()` was called without any arguments.

  The fix is always to push the NDB error as a warning and then set an error with a provided message. A new helper function has been added to `Thd_ndb` to be used in place of `ndb_my_error()`. (Bug #92244, Bug #28575934)
- Running out of undo log buffer memory was reported using error 921 `Out of transaction memory ... (increase SharedGlobalMemory)`.

  This problem is fixed by introducing a new error code 923 `Out of undo buffer memory (increase UNDO_BUFFER_SIZE)`. (Bug #92125, Bug #28537319)
- When moving an `OperationRec` from the serial to the parallel queue, `Dbacc::startNext()` failed to update the `Operationrec::OP_ACC_LOCK_MODE` flag which is required to reflect the accumulated `OP_LOCK_MODE` of all previous operations in the parallel queue. This inconsistency in the ACC lock queues caused the scan lock takeover mechanism to fail, as it incorrectly concluded that a lock to take over was not held. The same failure caused an assert when aborting an operation that was a member of such an inconsistent parallel lock queue. (Bug #92100, Bug #28530928)
- `ndb_restore` did not free all memory used after being called to restore a table that already existed. (Bug #92085, Bug #28525898)
- A data node failed during startup due to the arrival of a `SCAN_FRAGREQ` signal during the restore phase. This signal originated from a scan begun before the node had previously failed and which should have been aborted due to the involvement of the failed node in it. (Bug #92059, Bug #28518448)
- `DBTUP` sent the error `Tuple corruption detected` when a read operation attempted to read the value of a tuple inserted within the same transaction. (Bug #92009, Bug #28500861)

  References: See also: Bug #28893633.
- False constraint violation errors could occur when executing updates on self-referential foreign keys. (Bug #91965, Bug #28486390)

References: See also: Bug #90644, Bug #27930382.

- An NDB internal trigger definition could be dropped while pending instances of the trigger remained to be executed, by attempting to look up the definition for a trigger which had already been released. This caused unpredictable and thus unsafe behavior possibly leading to data node failure. The root cause of the issue lay in an invalid assumption in the code relating to determining whether a given trigger had been released; the issue is fixed by ensuring that the behavior of NDB, when a trigger definition is determined to have been released, is consistent, and that it meets expectations. (Bug #91894, Bug #28451957)

- In some cases, a workload that included a high number of concurrent inserts caused data node failures when using debug builds. (Bug #91764, Bug #28387450, Bug #29055038)

- During an initial node restart with disk data tables present and TwoPassInitialNodeRestartCopy enabled, DBTUP used an unsafe scan in disk order. Such scans are no longer employed in this case. (Bug #91724, Bug #28378227)

- Checking for old LCP files tested the table version, but this was not always dependable. Now, instead of relying on the table version, the check regards as invalid any LCP file having a maxGCI smaller than its createGci. (Bug #91637, Bug #28346565)

- In certain cases, a cascade update trigger was fired repeatedly on the same record, which eventually consumed all available concurrent operations, leading to Error 233 Out of operation records in transaction coordinator (increase MaxNoOfConcurrentOperations). If MaxNoOfConcurrentOperations was set to a value sufficiently high to avoid this, the issue manifested as data nodes consuming very large amounts of CPU, very likely eventually leading to a timeout. (Bug #91472, Bug #28262259)

Changes in MySQL NDB Cluster 8.0.13 (2018-10-23, Development Milestone)

MySQL NDB Cluster 8.0.13 is a new development release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in MySQL NDB Cluster 8.0.

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 8.0 through MySQL 8.0.13 (see Changes in MySQL 8.0.13 (2018-10-22, General Availability)).

- Functionality Added or Changed

- Bugs Fixed

Functionality Added or Changed

- **Important Change; NDB Disk Data:** The following changes are made in the display of information about Disk Data files in the INFORMATION_SCHEMA.FILES table:

  - Tablespaces and log file groups are no longer represented in the FILES table. (These constructs are not actually files.)

  - Each data file is now represented by a single row in the FILES table. Each undo log file is also now represented in this table by one row only. (Previously, a row was displayed for each copy of each of these files on each data node.)

- For rows corresponding to data files or undo log files, node ID and undo log buffer information is no longer displayed in the `EXTRA` column of the `FILES` table.

  Important

  The removal of undo log buffer information is reverted in NDB 8.0.15. (Bug #92796, Bug #28800252)

  (WL #11553)

- **Important Change; NDB Client Programs:** Removed the deprecated `--ndb` option for `perror`. Use `ndb_perror` to obtain error message information from `NDB` error codes instead. (Bug #81705, Bug #23523957)

  References: See also: Bug #81704, Bug #23523926.

- **Important Change:** Beginning with this release, MySQL NDB Cluster is being developed in parallel with the standard MySQL 8.0 server under a new unified release model with the following features:

  - NDB 8.0 is developed in, built from, and released with the MySQL 8.0 source code tree.
  - The numbering scheme for NDB Cluster 8.0 releases follows the scheme for MySQL 8.0, starting with the current MySQL release (8.0.13).
  - Building the source with NDB support appends `-cluster` to the version string returned by `mysql -V`, as shown here:

    ```
    shell≫ mysql -V
    mysql  Ver 8.0.13-cluster for Linux on x86_64 (Source distribution)
    ```

    NDB binaries continue to display both the MySQL Server version and the NDB engine version, like this:

    ```
    shell> ndb_mgm -V
    MySQL distrib mysql-8.0.13 ndb-8.0.13-dmr, for Linux (x86_64)
    ```

    In MySQL Cluster NDB 8.0, these two version numbers are always the same.

  To build the MySQL 8.0.13 (or later) source with NDB Cluster support, use the CMake option `-DWITH_NDBCLUSTER`. (WL #11762)

- **NDB Cluster APIs:** Added the `Table` methods `getExtraMetadata()` and `setExtraMetadata()`. (WL #9865)

- `INFORMATION_SCHEMA` tables now are populated with tablespace statistics for MySQL Cluster tables. (Bug #27167728)

- It is now possible to specify a set of cores to be used for I/O threads performing offline multithreaded builds of ordered indexes, as opposed to normal I/O duties such as file I/O , compression , or decompression. “Offline” in this context refers to building of ordered indexes performed when the parent table is not being written to; such building takes place when an `NDB` cluster performs a node or system restart, or as part of restoring a cluster from backup using `ndb_restore --rebuild-indexes`.

  In addition, the default behaviour for offline index build work is modified to use all cores available to `ndbmtd`, rather limiting itself to the core reserved for the I/O thread. Doing so can improve restart and restore times and performance, availability, and the user experience.

  This enhancement is implemented as follows:

  1. The default value for `BuildIndexThreads` is changed from 0 to 128. This means that offline ordered index builds are now multithreaded by default.

2. The default value for `TwoPassInitialNodeRestartCopy` is changed from `false` to `true`. This means that an initial node restart first copies all data from a “live” node to one that is starting —without creating any indexes—builds ordered indexes offline, and then again synchronizes its data with the live node, that is, synchronizing twice and building indexes offline between the two synchonizations. This causes an initial node restart to behave more like the normal restart of a node, and reduces the time required for building indexes.

3. A new thread type (`idxbld`) is defined for the `ThreadConfig` configuration parameter, to allow locking of offline index build threads to specific CPUs.

In addition, `NDB` now distinguishes the thread types that are accessible to “ThreadConfig” by the following two criteria:

1. Whether the thread is an execution thread. Threads of types `main`, `ldm`, `recv`, `rep`, `tc`, and `send` are execution threads; thread types `io`, `watchdog`, and `idxbld` are not.

2. Whether the allocation of the thread to a given task is permanent or temporary. Currently all thread types except `idxbld` are permanent.

For additonal information, see the descriptions of the parameters in the Manual. (Bug #25835748, Bug #26928111)

- When performing an NDB backup, the `ndbinfo.logbuffers` table now displays information regarding buffer usage by the backup process on each data node. This is implemented as rows reflecting two new log types in addition to `REDO` and `DD-UNDO`. One of these rows has the log type `BACKUP-DATA`, which shows the amount of data buffer used during backup to copy fragments to backup files. The other row has the log type `BACKUP-LOG`, which displays the amount of log buffer used during the backup to record changes made after the backup has started. One each of these `log_type` rows is shown in the `logbuffers` table for each data node in the cluster. Rows having these two log types are present in the table only while an NDB backup is currently in progress. (Bug #25822988)

- Added the `ODirectSyncFlag` configuration parameter for data nodes. When enabled, the data node treats all completed filesystem writes to the redo log as though they had been performed using `fsync`.

  Note

  This parameter has no effect if at least one of the following conditions is true:

  - `ODirect` is not enabled.
  - `InitFragmentLogFiles` is set to `SPARSE`.

  (Bug #25428560)

- Added the `--logbuffer-size` option for `ndbd` and `ndbmtd`, for use in debugging with a large number of log messages. This controls the size of the data node log buffer; the default (32K) is intended for normal operations. (Bug #89679, Bug #27550943)

- Prior to NDB 8.0, all string hashing was based on first transforming the string into a normalized form, then MD5-hashing the resulting binary image. This could give rise to some performance problems, for the following reasons:

  - The normalized string is always space padded to its full length. For a `VARCHAR`, this often involved adding more spaces than there were characters in the original string.

  - The string libraries were not optimized for this space padding, and added considerable overhead in some use cases.

  - The padding semantics varied between character sets, some of which were not padded to their full length.

  - The transformed string could become quite large, even without space padding; some Unicode 9.0 collations can transform a single code point into 100 bytes of character data or more.

  - Subsequent MD5 hashing consisted mainly of padding with spaces, and was not particularly efficient, possibly causing additional performance penalties by flush significant portions of the L1 cache.

  Collations provide their own hash functions, which hash the string directly without first creating a normalized string. In addition, for Unicode 9.0 collations, the hashes are computed without padding. `NDB` now takes advantage of this built-in function whenever hashing a string identified as using a Unicode 9.0 collation.

  Since, for other collations there are existing databases which are hash partitioned on the transformed string, `NDB` continues to employ the previous method for hashing strings that use these, to maintain compatibility. (Bug #89609, Bug #27523758)

  References: See also: Bug #89590, Bug #27515000, Bug #89604, Bug #27522732.

- A table created in NDB 7.6 and earlier contains metadata in the form of a compressed `.frm` file, which is no longer supported in MySQL 8.0. To facilitate online upgrades to NDB 8.0, `NDB` performs on-the-fly translation of this metadata and writes it into the MySQL Server's data dictionary, which enables the `mysqld` in NDB Cluster 8.0 to work with the table without preventing subsequent use of the table by a previous version of the `NDB` software.

  Important

  Once a table's structure has been modified in NDB 8.0, its metadata is stored using the Data Dictionary, and it can no longer be accessed by NDB 7.6 and earlier.

  This enhancement also makes it possible to restore an `NDB` backup made using an earlier version to a cluster running NDB 8.0 (or later). (WL #10167)

Bugs Fixed

- **Important Change; NDB Disk Data:** It was possible to issue a `CREATE TABLE` statement that referred to a nonexistent tablespace. Now such a statement fails with an error. (Bug #85859, Bug #25860404)

- **Important Change; NDB Replication:** Because the MySQL Server now executes `RESET MASTER` with a global read lock, the behavior of this statement when used with NDB Cluster has changed in the following two respects:

  - It is no longer guaranteed to be synchronous; that is, it is now possible that a read coming immediately before `RESET MASTER` is issued may not be logged until after the binary log has been rotated.

- It now behaves identically, regardless of whether the statement is issued on the same SQL node that is writing the binary log, or on a different SQL node in the same cluster.

  Note

  `SHOW BINLOG EVENTS`, `FLUSH LOGS`, and most data definition statements continue, as they did in previous `NDB` versions, to operate in a synchronous fashion.

  (Bug #89976, Bug #27665565)

- **Important Change:** `NDB` supports any of the following three values for the `CREATE TABLE` statement's `ROW_FORMAT` option: `DEFAULT`, `FIXED`, and `DYNAMIC`. Formerly, any values other than these were accepted but resulted in `DYNAMIC` being used. Now a `CREATE TABLE` statement that attempts to create an `NDB` table fails with an error if `ROW_FORMAT` is used, and does not have one of the three values listed. (Bug #88803, Bug #27230898)

- **Microsoft Windows; ndbinfo Information Database:** The process ID of the monitor process used on Windows platforms by `RESTART` to spawn and restart a `mysqld` is now shown in the `ndbinfo.processes` table as an `angel_pid`. (Bug #90235, Bug #27767237)

- **NDB Cluster APIs:** The example NDB API programs `ndbapi_array_simple` and `ndbapi_array_using_adapter` did not perform cleanup following execution; in addition, the example program `ndbapi_simple_dual` did not check to see whether the table used by this example already existed. Due to these issues, none of these examples could be run more than once in succession.

  The issues just described have been corrected in the example sources, and the relevant code listings in the NDB API documentation have been updated to match. (Bug #27009386)

- **NDB Cluster APIs:** A previous fix for an issue, in which the failure of multiple data nodes during a partial restart could cause API nodes to fail, did not properly check the validity of the associated `NdbReceiver` object before proceeding. Now in such cases an invalid object triggers handling for invalid signals, rather than a node failure. (Bug #25902137)

  References: This issue is a regression of: Bug #25092498.

- **NDB Cluster APIs:** Incorrect results, usually an empty result set, were returned when `setBound()` was used to specify a `NULL` bound. This issue appears to have been caused by a problem in gcc, limited to cases using the old version of this method (which does not employ `NdbRecord`), and is fixed by rewriting the problematic internal logic in the old implementation. (Bug #89468, Bug #27461752)

- **NDB Cluster APIs:** Released NDB API objects are kept in one or more `Ndb_free_list` structures for later reuse. Each list also keeps track of all objects seized from it, and makes sure that these are eventually released back to it. In the event that the internal function `NdbScanOperation::init()` failed, it was possible for an `NdbApiSignal` already allocated by the `NdbOperation` to be leaked. Now in such cases, `NdbScanOperation::release()` is called to release any objects allocated by the failed `NdbScanOperation` before it is returned to the free list.

  This fix also handles a similar issue with `NdbOperation::init()`, where a failed call could also leak a signal. (Bug #89249, Bug #27389894)

- **NDB Cluster APIs:** Removed the unused `TFSentinel` implementation class, which raised compiler warnings on 32-bit systems. (Bug #89005, Bug #27302881)

- **NDB Cluster APIs:** The success of thread creation by API calls was not always checked, which could lead to timeouts in some cases. (Bug #88784, Bug #27225714)

- **NDB Cluster APIs:** The file `storage/ndb/src/ndbapi/ndberror.c` was renamed to `ndberror.cpp`. (Bug #87725, Bug #26781567)

- **ndbinfo Information Database:** Counts of committed rows and committed operations per fragment used by some tables in `ndbinfo` were taken from the `DBACC` block, but due to the fact that commit signals can arrive out of order, transient counter values could be negative. This could happen if, for example, a transaction contained several interleaved insert and delete operations on the same row; in such cases, commit signals for delete operations could arrive before those for the corresponding insert operations, leading to a failure in `DBACC`.

  This issue is fixed by using the counts of committed rows which are kept in `DBTUP`, which do not have this problem. (Bug #88087, Bug #26968613)

- **NDB Client Programs:** When passed an invalid connection string, the `ndb_mgm` client did not always free up all memory used before exiting. (Bug #90179, Bug #27737906)

- **NDB Client Programs:** `ndb_show_tables` did not always free up all memory which it used. (Bug #90152, Bug #27727544)

- **NDB Client Programs:** On Unix platforms, the Auto-Installer failed to stop the cluster when `ndb_mgmd` was installed in a directory other than the default. (Bug #89624, Bug #27531186)

- **NDB Client Programs:** The Auto-Installer did not provide a mechanism for setting the `ServerPort` parameter. (Bug #89623, Bug #27539823)

- **MySQL NDB ClusterJ:** When a table containing a `BLOB` or a `TEXT` field was being queried with ClusterJ for a record that did not exist, an exception ("`The method is not valid in current blob state`") was thrown. (Bug #28536926)

- **MySQL NDB ClusterJ:** ClusterJ quit unexpectedly as there was no error handling in the `scanIndex()` function of the `ClusterTransactionImpl` class for a null returned to it internally by the `scanIndex()` method of the `ndbTransaction` class. (Bug #27297681, Bug #88989)

- Local checkpoints did not always handle `DROP TABLE` operations correctly. (Bug #27926532)

  References: This issue is a regression of: Bug #26908347, Bug #26968613.

- In some circumstances, when a transaction was aborted in the `DBTC` block, there remained links to trigger records from operation records which were not yet reference-counted, but when such an operation record was released the trigger reference count was still decremented. (Bug #27629680)

- An internal buffer being reused immediately after it had been freed could lead to an unplanned data node shutdown. (Bug #27622643)

  References: See also: Bug #28698831.

- An `NDB` online backup consists of data, which is fuzzy, and a redo and undo log. To restore to a consistent state it is necessary to ensure that the log contains all of the changes spanning the capture of the fuzzy data portion and beyond to a consistent snapshot point. This is achieved by waiting for a GCI boundary to be passed after the capture of data is complete, but before stopping change logging and recording the stop GCI in the backup's metadata.

  At restore time, the log is replayed up to the stop GCI, restoring the system to the state it had at the consistent stop GCI. A problem arose when, under load, it was possible to select a GCI boundary which occurred too early and did not span all the data captured. This could lead to inconsistencies when restoring the backup; these could be noticed as broken constraints or corrupted `BLOB` entries.

  Now the stop GCI is chosen is so that it spans the entire duration of the fuzzy data capture process, so that the backup log always contains all data within a given stop GCI. (Bug #27497461)

  References: See also: Bug #27566346.

- For `NDB` tables, when a foreign key was added or dropped as a part of a DDL statement, the foreign key metatdata for all parent tables referenced should be reloaded in the handler on all SQL

nodes connected to the cluster, but this was done only on the `mysqld` on which the statement was executed. Due to this, any subsequent queries relying on foreign key metadata from the corresponding parent tables could return inconsistent results. (Bug #27439587)

References: See also: Bug #82989, Bug #24666177.

- `ANALYZE TABLE` used excessive amounts of CPU on large, low-cardinality tables. (Bug #27438963)
- Queries using very large lists with `IN` were not handled correctly, which could lead to data node failures. (Bug #27397802)

  References: See also: Bug #28728603.
- A data node overload could in some situations lead to an unplanned shutdown of the data node, which led to all data nodes disconnecting from the management and nodes.

  This was due to a situation in which `API_FAILREQ` was not the last received signal prior to the node failure.

  As part of this fix, the transaction coordinator's handling of `SCAN_TABREQ` signals for an `ApiConnectRecord` in an incorrect state was also improved. (Bug #27381901)

  References: See also: Bug #47039, Bug #11755287.
- In a two-node cluster, when the node having the lowest ID was started using `--nostart`, API clients could not connect, failing with `Could not alloc node id at HOST port PORT_NO: No free node id found for mysqld(API)`. (Bug #27225212)
- Changing `MaxNoOfExecutionThreads` without an initial system restart led to an unplanned data node shutdown. (Bug #27169282)

  References: This issue is a regression of: Bug #26908347, Bug #26968613.
- In most cases, and especially in error conditions, `NDB` command-line programs failed on exit to free memory used by option handling, and failed to call `ndb_end()`. This is fixed by removing the internal methods `ndb_load_defaults()` and `ndb_free_defaults()` from `storage/ndb/include/util/ndb_opts.h`, and replacing these with an `Ndb_opts` class that automatically frees such resources as part of its destructor. (Bug #26930148)

  References: See also: Bug #87396, Bug #26617328.
- A query against the `INFORMATION_SCHEMA.FILES` table returned no results when it included an `ORDER BY` clause. (Bug #26877788)
- ClusterJ failed to connect to a MySQL node that used utf8mb4_800_ci_ai as its default character set for connection. Also, ClusterJ quit unexpectedly when handling a table with a character set number of 255 or larger. This fix corrected both issues. (Bug #26027722)
- During a restart, `DBLQH` loads redo log part metadata for each redo log part it manages, from one or more redo log files. Since each file has a limited capacity for metadata, the number of files which must be consulted depends on the size of the redo log part. These files are opened, read, and closed sequentially, but the closing of one file occurs concurrently with the opening of the next.

  In cases where closing of the file was slow, it was possible for more than 4 files per redo log part to be open concurrently; since these files were opened using the `OM_WRITE_BUFFER` option, more than 4 chunks of write buffer were allocated per part in such cases. The write buffer pool is not unlimited; if all redo log parts were in a similar state, the pool was exhausted, causing the data node to shut down.

  This issue is resolved by avoiding the use of `OM_WRITE_BUFFER` during metadata reload, so that any transient opening of more than 4 redo log files per log file part no longer leads to failure of the data node. (Bug #25965370)

- Under certain conditions, data nodes restarted unnecessarily during execution of `ALTER TABLE... REORGANIZE PARTITION`. (Bug #25675481)

  References: See also: Bug #26735618, Bug #27191468.

- Race conditions sometimes occurred during asynchronous disconnection and reconnection of the transporter while other threads concurrently inserted signal data into the send buffers, leading to an unplanned shutdown of the cluster.

  As part of the work fixing this issue, the internal templating function used by the Transporter Registry when it prepares a send is refactored to use likely-or-unlikely logic to speed up execution, and to remove a number of duplicate checks for NULL. (Bug #24444908, Bug #25128512)

  References: See also: Bug #20112700.

- `ndb_restore` sometimes logged data file and log file progress values much greater than 100%. (Bug #20989106)

- Removed unneeded debug printouts from the internal function `ha_ndbcluster::copy_fk_for_offline_alter()`. (Bug #90991, Bug #28069711)

- The internal function `BitmaskImpl::setRange()` set one bit fewer than specified. (Bug #90648, Bug #27931995)

- Inserting a row into an `NDB` table having a self-referencing foreign key that referenced a unique index on the table other than the primary key failed with `ER_NO_REFERENCED_ROW_2`. This was due to the fact that `NDB` checked foreign key constraints before the unique index was updated, so that the constraint check was unable to use the index for locating the row. Now, in such cases, `NDB` waits until all unique index values have been updated before checking foreign key constraints on the inserted row. (Bug #90644, Bug #27930382)

  References: See also: Bug #91965, Bug #28486390.

- Removed all references to the C++ `register` storage class in the NDB Cluster sources; use of this specifier, which was deprecated in C++11 and removed in C++17, raised warnings when building with recent compilers. (Bug #90110, Bug #27705985)

- It was not possible to create an `NDB` table using `PARTITION_BALANCE` set to `FOR_RA_BY_LDM_X_2`, `FOR_RA_BY_LDM_X_3`, or `FOR_RA_BY_LDM_X_4`. (Bug #89811, Bug #27602352)

  References: This issue is a regression of: Bug #81759, Bug #23544301.

- Adding a `[tcp]` or `[shm]` section to the global configuration file for a cluster with multiple data nodes caused default TCP connections to be lost to the node using the single section. (Bug #89627, Bug #27532407)

- Removed a memory leak in the configuration file parser. (Bug #89392, Bug #27440614)

- Fixed a number of implicit-fallthrough warnings, warnings raised by uninitialized values, and other warnings encountered when compiling `NDB` with GCC 7.2.0. (Bug #89254, Bug #89255, Bug #89258, Bug #89259, Bug #89270, Bug #27390714, Bug #27390745, Bug #27390684, Bug #27390816, Bug #27396662)

  References: See also: Bug #88136, Bug #26990244.

- Node connection states were not always reported correctly by `ClusterMgr` immediately after reporting a disconnection. (Bug #89121, Bug #27349285)

- As a result of the reuse of code intended for send threads when performing an assist send, all of the local release send buffers were released to the global pool, which caused the intended level of the local send buffer pool to be ignored. Now send threads and assisting worker threads follow their own policies for maintaining their local buffer pools. (Bug #89119, Bug #27349118)

- When the PGMAN block seized a new Page_request record using seizeLast, its return value was not checked, which could cause access to invalid memory. (Bug #89009, Bug #27303191)
- TCROLLBACKREP signals were not handled correctly by the DBTC kernel block. (Bug #89004, Bug #27302734)
- When sending priority A signals, we now ensure that the number of pending signals is explicitly initialized. (Bug #88986, Bug #27294856)
- The internal function ha_ndbcluster::unpack_record() did not perform proper error handling. (Bug #88587, Bug #27150980)
- CHECKSUM is not supported for NDB tables, but this was not not reflected in the CHECKSUM column of the INFORMATION_SCHEMA.TABLES table, which could potentially assume a random value in such cases. Now the value of this column is always set to NULL for NDB tables, just as it is for InnoDB tables. (Bug #88552, Bug #27143813)
- Removed a memory leak detected when running ndb_mgm -e "CLUSTERLOG ...". (Bug #88517, Bug #27128846)
- When terminating, ndb_config did not release all memory which it had used. (Bug #88515, Bug #27128398)
- ndb_restore did not free memory properly before exiting. (Bug #88514, Bug #27128361)
- In certain circumstances where multiple Ndb objects were being used in parallel from an API node, the block number extracted from a block reference in DBLQH was the same as that of a SUMA block even though the request was coming from an API node. Due to this ambiguity, DBLQH mistook the request from the API node for a request from a SUMA block and failed. This is fixed by checking node IDs before checking block numbers. (Bug #88441, Bug #27130570)
- A join entirely within the materialized part of a semijoin was not pushed even if it could have been. In addition, EXPLAIN provided no information about why the join was not pushed. (Bug #88224, Bug #27022925)

  References: See also: Bug #27067538.
- All known compiler warnings raised by -Werror when building the NDB source code have been fixed. (Bug #88136, Bug #26990244)
- When the duplicate weedout algorithm was used for evaluating a semijoin, the result had missing rows. (Bug #88117, Bug #26984919)

  References: See also: Bug #87992, Bug #26926666.
- NDB did not compile with GCC 7. (Bug #88011, Bug #26933472)
- A table used in a loose scan could be used as a child in a pushed join query, leading to possibly incorrect results. (Bug #87992, Bug #26926666)
- When representing a materialized semijoin in the query plan, the MySQL Optimizer inserted extra QEP_TAB and JOIN_TAB objects to represent access to the materialized subquery result. The join pushdown analyzer did not properly set up its internal data structures for these, leaving them uninitialized instead. This meant that later usage of any item objects referencing the materialized semijoin accessed an initialized tableno column when accessing a 64-bit tableno bitmask, possibly referring to a point beyond its end, leading to an unplanned shutdown of the SQL node. (Bug #87971, Bug #26919289)
- In some cases, a SCAN_FRAGCONF signal was received after a SCAN_FRAGREQ with a close flag had already been sent, clearing the timer. When this occurred, the next SCAN_FRAGREF to arrive caused time tracking to fail. Now in such cases, a check for a cleared timer is performed prior to processing the SCAN_FRAGREF message. (Bug #87942, Bug #26908347)

- While deleting an element in `Dbacc`, or moving it during hash table expansion or reduction, the method used (`getLastAndRemove()`) could return a reference to a removed element on a released page, which could later be referenced from the functions calling it. This was due to a change brought about by the implementation of dynamic index memory in NDB 7.6.2; previously, the page had always belonged to a single `Dbacc` instance, so accessing it was safe. This was no longer the case following the change; a page released in `Dbacc` could be placed directly into the global page pool where any other thread could then allocate it.

  Now we make sure that newly released pages in `Dbacc` are kept within the current `Dbacc` instance and not given over directly to the global page pool. In addition, the reference to a released page has been removed; the affected internal method now returns the last element by value, rather than by reference. (Bug #87932, Bug #26906640)

  References: See also: Bug #87987, Bug #26925595.

- When creating a table with a nonexistent conflict detection function, `NDB` returned an improper error message. (Bug #87628, Bug #26730019)

- `ndb_top` failed to build with the error `"HAVE_NCURSESW_H" is not defined`. (Bug #87035, Bug #26429281)

- In a MySQL Cluster with one MySQL Server configured to write a binary log failure occurred when creating and using an `NDB` table with non-stored generated columns. The problem arose only when the product was built with debugging support. (Bug #86084, Bug #25957586)

- It was possible to create or alter a `STORAGE MEMORY` table using a nonexistent tablespace without any error resulting. Such an operation now fails with Error 3510 `ER_TABLESPACE_MISSING_WITH_NAME`, as intended. (Bug #82116, Bug #23744378)

- `ndb_restore --print-data --hex` did not print trailing 0s of `LONGVARBINARY` values. (Bug #65560, Bug #14198580)

- When the internal function `ha_ndbcluster::copy_fk_for_offline_alter()` checked dependent objects on a table from which it was supposed to drop a foreign key, it did not perform any filtering for foreign keys, making it possible for it to attempt retrieval of an index or trigger instead, leading to a spurious Error 723 (`No such table`).

Release Series Changelogs: MySQL NDB Cluster 8.0

This section contains unified changelog information for the NDB Cluster 8.0 release series.

For changelogs covering individual MySQL NDB Cluster 8.0 releases, see NDB Cluster Release Notes.

For general information about features added in MySQL NDB Cluster 8.0, see What is New in NDB Cluster 8.0.

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

Changes in MySQL NDB Cluster 8.0.38 (2024-07-02, General Availability)

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 8.0.39 instead.

- Functionality Added or Changed

- Bugs Fixed

Functionality Added or Changed

- **ndbinfo Information Database:** The following columns have been added to the `transporter_details` table:

  - `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.

  - `type`: The connection type used by this transporter (`TCP` or `SHM`).

  See The ndbinfo transporter_details Table, for more information. (Bug #36579842)

  References: See also: Bug #36569947.

- **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 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 from `src/ndbapi/NdbDictionaryImpl.cpp`. (Bug #36532102)

- 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)

- An error injected in the `LQH` proxy and then resent to all workers was eventually cleared by the workers, but not in the `LQH` proxy where it was never cleared. (Bug #36398772)

- 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)

Changes in MySQL NDB Cluster 8.0.37 (2024-04-30, General Availability)

- 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 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 Client Programs:** `ndb_redo_log_reader` could not read data from encrypted files. (Bug #36313482)
- **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:** `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:** Invoking ndb_mgmd with the --bind-address option could in some cases cause the program to terminate unexpectedly. (Bug #36263410)
- **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 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)

- `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)

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

- 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)
- 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.
- 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 8.0.36 (2024-01-16, General Availability)

- Compilation Notes
- Bugs Fixed

Compilation Notes

- 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)

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.
- **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)
- **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)
- 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.

- 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)
- 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.
- 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)
- 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)

- 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)

- 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 8.0.35 (2023-10-25, General Availability)

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:** `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)

- 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)

- 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)

- 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.

- 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.

- 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.

- 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)

Changes in MySQL NDB Cluster 8.0.34 (2023-07-18, General Availability)

- 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 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.

- 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)

Changes in MySQL NDB Cluster 8.0.33 (2023-04-18, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **NDB Cluster APIs:** The `Node.js` library used to build the MySQL NoSQL Connector for JavaScript has been upgraded to version 18.12.1. (Bug #35095122)
- **MySQL NDB ClusterJ:** Performance has been improved for accessing tables using a single-column partition key when the column is of type `CHAR` or `VARCHAR`. (Bug #35027961)
- Beginning with this release, `ndb_restore` implements the `--timestamp-printouts` option, which causes all error, info, and debug node log messages to be prefixed with timestamps. (Bug #34110068)

Bugs Fixed

- **Microsoft Windows:** Two memory leaks found by code inspection were removed from `NDB` process handles on Windows platforms. (Bug #34872901)
- **Microsoft Windows:** On Windows platforms, the data node angel process did not detect whether a child data node process exited normally. We fix this by keeping an open process handle to the child and using this when probing for the child's exit. (Bug #34853213)
- **NDB Cluster APIs; MySQL NDB ClusterJ:** MySQL ClusterJ uses a scratch buffer for primary key hash calculations which was limited to 10000 bytes, which proved too small in some cases. Now we `malloc()` the buffer if its size is not sufficient.

  This also fixes an issue with the `Ndb` object methods `startTransaction()` and `computeHash()` in the NDB API: Previously, if either of these methods was passed a temporary buffer of insufficient size, the method failed. Now in such cases a temporary buffer is allocated.

  Our thanks to Mikael Ronström for this contribution. (Bug #103814, Bug #32959894)

- **NDB Cluster APIs:** When dropping an event operation (`NdbEventOperation`) in the NDB API, it was sometimes possible for the dropped event operation to remain visible to the application after instructing the data nodes to stop sending events related to this event operation, but before all pending buffered events were consumed and discarded. This could be observed in certain cases when performing an online alter operation, such as `ADD COLUMN` or `RENAME COLUMN`, along with concurrent writes to the affected table.

Further analysis showed that the dropped events were accessible when iterating through event operations with `Ndb::getGCIEventOperations()`. Now, this method skips dropped events when called iteratively. (Bug #34809944)

- **NDB Cluster APIs:** `Event::getReport()` always returned an error for an event opened from `NDB`, instead of returning the flags actually used by the report object. (Bug #34667384)

- Before a new `NDB` table definition can be stored in the data dictionary, any existing definition must be removed. Table definitions have two unique values, the table name and the NDB Cluster `se_private_id`. During installation of a new table definition, we check whether there is any existing definition with the same table name and, if so, remove it. Then we check whether the table removed and the one being installed have the same `se_private_id`; if they do not, any definition that is occupying this `se_private_id` is considered stale, and removed as well.

  Problems arose when no existing definition was found by the search using the table's name, since no definition was dropped even if one occupied `se_private_id`, leading to a duplicate key error when attempting to store the new table. The internal `store_table()` function attempted to clear the diagnostics area, remove the stale definition of `se_private_id`, and try to store it once again, but the diagnostics area was not actually cleared, thus leaking the error is thus leaked and presenting it to the user.

  To fix this, we remove any stale table definition, regardless of any action taken (or not) by `store_table()`. (Bug #35089015)

- Fixed the following two issues in the output of `ndb_restore`:

  - The backup file format version was shown for both the backup file format version and the version of the cluster which produced the backup.

  - To reduce confusion between the version of the file format and the version of the cluster which produced the backup, the backup file format version is now shown using hexadecimal notation.

  (Bug #35079426)

  References: This issue is a regression of: Bug #34110068.

- Removed a memory leak in the `DBDICT` kernel block caused when an internal foreign key definition record was not released when no longer needed. This could be triggered by either of the following events:

  - Drop of a foreign key constraint on an `NDB` table

  - Rejection of an attempt to create a foreign key constraint on an `NDB` table

  Such records use the `DISK_RECORDS` memory resource; you can check this on a running cluster by executing `SELECT node_id, used FROM ndbinfo.resources WHERE resource_name='DISK_RECORDS'` in the `mysql` client. This resource uses `SharedGlobalMemory`, exhaustion of which could lead not only to the rejection of attempts to create foreign keys, but of queries making use of joins as well, since the `DBSPJ` block also uses shared global memory by way of `QUERY_MEMORY`. (Bug #35064142)

- When attempting a copying alter operation with `--ndb-allow-copying-alter-table = OFF`, the reason for rejection of the statement was not always made clear to the user. (Bug #35059079)

- When a transaction coordinator is starting fragment scans with many fragments to scan, it may take a realtime break (RTB) during the process to ensure fair CPU access for other requests. When the requesting API disconnected and API failure handling for the scan state occurred before the RTB

continuation returned, continuation processing could not proceed because the scan state had been removed.

We fix this by adding appropriate checks on the scan state as part of the continuation process. (Bug #35037683)

- Sender and receiver signal IDs were printed in trace logs as signed values even though they are actually unsigned 32-bit numbers. This could result in confusion when the top bit was set, as it cuased such numbers to be shown as negatives, counting upwards from `-MAX_32_BIT_SIGNED_INT`. (Bug #35037396)

- A fiber used by the `DICT` block monitors all indexes, and triggers index statistics calculations if requested by `DBTUX` index fragment monitoring; these calculations are performed using a schema transaction. When the `DICT` fiber attempts but fails to seize a transaction handle for requesting a schema transaction to be started, fiber exited, so that no more automated index statistics updates could be performed without a node failure. (Bug #34992370)

  References: See also: Bug #34007422.

- Schema objects in NDB use composite versioning, comprising major and minor subversions. When a schema object is first created, its major and minor versions are set; when an existing schema object is altered in place, its minor subversion is incremented.

  At restart time each data node checks schema objects as part of recovery; for foreign key objects, the versions of referenced parent and child tables (and indexes, for foreign key references not to or from a table's primary key) are checked for consistency. The table version of this check compares only major subversions, allowing tables to evolve, but the index version also compares minor subversions; this resulted in a failure at restart time when an index had been altered.

  We fix this by comparing only major subversions for indexes in such cases. (Bug #34976028)

  References: See also: Bug #21363253.

- `ndb_import` sometimes silently ignored hint failure for tables having large `VARCHAR` primary keys. For hinting which transaction coordinator to use, `ndb_import` can use the row's partitioning key, using a 4092 byte buffer to compute the hash for the key.

  This was problematic when the key included a `VARCHAR` column using UTF8, since the hash buffer may require in bytes up to 24 times the number of maximum characters in the column, depending on the column's collation; the hash computation failed but the calling code in `ndb_import` did not check for this, and continued using an undefined hash value which yielded an undefined hint.

  This did not lead to any functional problems, but was not optimal, and the user was not notified of it.

  We fix this by ensuring that `ndb_import` always uses sufficient buffer for handling character columns (regardless of their collations) in the key, and adding a check in `ndb_import` for any failures in hash computation and reporting these to the user. (Bug #34917498)

- When the `ndbcluster` plugin creates the `ndb_schema` table, the plugin inserts a row containing metadata, which is needed to keep track of this NDB Cluster instance, and which is stored as a set of key-value pairs in a row in this table.

  The `ndb_schema` table is hidden from MySQL and so not possible to query using SQL, but contains a UUID generated by the same MySQL server that creates the `ndb_schema` table; the same UUID is also stored as metadata in the data dictionary of each MySQL Server when the `ndb_schema` table is installed on it.

  When a `mysqld` connects (or reconnects) to `NDB`, it compares the UUID in its own data dictionary with the UUID stored in `NDB` in order to detect whether it is reconnecting to the same cluster; if not, the entire contents of the data dictionary are scrapped in order to make it faster and easier to install all tables fresh from `NDB`.

One such case occurs when all NDB data nodes have been restarted with --initial, thus removing all data and tables. Another happens when the ndb_schema table has been restored from a backup without restoring any of its data, since this means that the row for the ndb_schema table would be missing.

To deal with these types of situations, we now make sure that, when synchronization has completed, there is always a row in the NDB dictionary with a UUID matching the UUID stored in the MySQL server data dictionary. (Bug #34876468)

- When running an NDB Cluster with multiple management servers, termination of the ndb_mgmd processes required an excessive amount of time when shutting down the cluster. (Bug #34872372)
- Schema distribution timeout was detected by the schema distribution coordinator after dropping and re-creating the mysql.ndb_schema table when any nodes that were subscribed beforehand had not yet resubscribed when the next schema operation began. This was due to a stale list of subscribers being left behind in the schema distribution data; these subscribers were assumed by the coordinator to be participants in subsequent schema operations.

  We fix this issue by clearing the list of known subscribers whenever the mysql.ndb_schema table is dropped. (Bug #34843412)
- When requesting a new global checkpoint (GCP) from the data nodes, such as by the NDB Cluster handler in mysqld to speed up delivery of schema distribution events and responses, the request was sent 100 times. While the DBDIH block attempted to merge these duplicate requests into one, it was possible on occasion to trigger more than one immediate GCP. (Bug #34836471)
- When the DBSPJ block receives a query for execution, it sets up its own internal plan for how to do so. This plan is based on the query plan provided by the optimizer, with adaptions made to provide the most efficient execution of the query, both in terms of elapsed time and of total resources used.

  Query plans received by DBSPJ often contain star joins, in which several child tables depend on a common parent, as in the query shown here:

  ```
  SELECT STRAIGHT_JOIN * FROM t AS t1
  INNER JOIN t AS t2 ON t2.a = t1.k
  INNER JOIN t AS t3 ON t3.k = t1.k;
  ```

  In such cases DBSPJ could submit key-range lookups to t2 and t3 in parallel (but does not do so). An inner join also has the property that each inner joined row requires a match from the other tables in the same join nest, else the row is eliminated from the result set. Thus, by using the key-range lookups, we may retrieve rows from one such lookup which have no matches in the other, which effort is ultimately wasted. Instead, DBSPJ sets up a sequential plan for such a query.

  It was found that this worked as intended for queries having only inner joins, but if any of the tables are left-joined, we did not take complete advantage of the preceding inner joined tables before issuing the outer joined tables. Suppose the previous query is modified to include a left join, like this:

  ```
  SELECT STRAIGHT_JOIN * FROM t AS t1
  INNER JOIN t AS t2 ON t2.a = t1.k
  LEFT  JOIN t AS t3 ON t3.k = t1.k;
  ```

  Using the following query against the ndbinfo.counters table, it is possible to observe how many rows are returned for each query before and after query execution:

  ```
  SELECT counter_name, SUM(val)
  FROM ndbinfo.counters
  WHERE block_name="DBSPJ" AND counter_name = "SCAN_ROWS_RETURNED";
  ```

  It was thus determined that requests on t2 and t3 were submitted in parallel. Now in such cases, we wait for the inner join to complete before issuing the left join, so that unmatched rows from t1 can be eliminated from the outer join on t1 and t3. This results in less work to be performed by the data nodes, and reduces the volumne handled by the transporter as well. (Bug #34782276)

- SPJ handling of a sorted result was found to suffer a significant performance impact compared to the same result set when not sorted. Further investigation showed that most of the additional performance overhead for sorted results lay in the implementation for sorted result retrieval, which required an excessive number of `SCAN_NEXTREQ` round trips between the client and `DBSPJ` on the data nodes. (Bug #34768353)

- `DBSPJ` now implements the `firstMatch` optimization for semijoins and antijoins, such as those found in `EXISTS` and `NOT EXISTS` subqueries. (Bug #34768191)

- When the `DBSPJ` block sends `SCAN_FRAGREQ` and `SCAN_NEXTREQ` signals to the data nodes, it tries to determine the optimum number of fragments to scan in parallel without starting more parallel scans than needed to fill the available batch buffers, thus avoiding any need to send additional `SCAN_NEXTREQ` signals to complete the scan of each fragment.

  The `DBSPJ` block's statistics module calculates and samples the parallelism which was optimal for fragment scans just completed, for each completed `SCAN_FRAGREQ`, providing a mean and standard deviation of the sampled parallelism. This makes it possible to calculate a lower 95th percentile of the parallelism (and batch size) which makes it possible to complete a `SCAN_FRAGREQ` without needing additional `SCAN_NEXTREQ` signals.

  It was found that the parallelism statistics seemed unable to provide a stable parallelism estimate and that the standard deviation was unexpectedly high. This often led to the parallelism estimate being a negative number (always rounded up to 1).

  The flaw in the statistics calculation was found to be an underlying assumption that each sampled `SCAN_FRAGREQ` contained the same number of key ranges to be scanned, which is not necessarily the case. Typically a full batch of rows for the first `SCAN_FRAGREQ`, and relatively few rows for the final `SCAN_NEXTREQ` returning the remaining rows; this resulted in wide variation in parallelism samples which made the statistics obtained from them unreliable.

  We fix this by basing the statistics on the number of keys actually sent in the `SCAN_FRAGREQ`, and counting the rows returned from this request. Based on this it is possible to obtain record-per-key statistics to be calculated and sampled. This makes it possible to calculate the number of fragments which can be scanned, without overflowing the batch buffers. (Bug #34768106)

- It was possible in certain cases that both the `NDB` binary logging thread and metadata synchronization attempted to synchronize the `ndb_apply_status` table, which led to a race condition. We fix this by making sure that the `ndb_apply_status` table is monitored and created (or re-created) by the binary logging thread only. (Bug #34750992)

- While starting a schema operation, the client is responsible for detecting timeouts until the coordinator has received the first schema event; from that point, any schema operation timeout should be detected by the coordinator. A problem occurred while the client was checking the timeout; it mistakenly set the state indicating that timeout had occurred, which caused the coordinator to ignore the first schema event taking longer than approximately one second to receive (that is, to write the send event plus handle in the binary logging thread). This had the effect that, in these cases, the coordinator was not involved in the schema operation.

  We fix this by change the schema distribution timeout checking to be atomic, and to let it be performed by either the client or the coordinator. In addition, we remove the state variable used for keeping track of events received by the coordinator, and rely on the list of participants instead. (Bug #34741743)

- An SQL node did not start up correctly after restoring data with `ndb_restore`, such that, when it was otherwise ready to accept connections, the binary log injector thread never became ready. It was found that, when a `mysqld` was started after a data node initial restore from which new table

IDs were generated, the utility table's (`ndb_*`) MySQL data dictionary definition might not match the NDB dictionary definition.

The existing `mysqld` definition is dropped by name, thus removing the unique `ndbcluster-ID` key in the MySQL data dictionary but the new table ID could also already be occupied by another (stale) definition. The resulting mistmatch prevented setup of the binary log.

To fix this problem we now explicitly drop any `ndbcluster-ID` definitions that might clash in such cases with the table being installed. (Bug #34733051)

- After receiving a `SIGTERM` signal, `ndb_mgmd` did not wait for all threads to shut down before exiting. (Bug #33522783)

  References: See also: Bug #32446105.

- When multiple operations are pending on a single row, it is not possible to commit an operation which is run concurrently with an operation which is pending abort. This could lead to data node shutdown during the commit operation in `DBACC`, which could manifest when a single transaction contained more than `MaxDMLOperationsPerTransaction` DML operations.

  In addition, a transaction containing insert operations is rolled back if a statement that uses a locking scan on the prepared insert fails due to too many DML operations. This could lead to an unplanned data node shutdown during tuple deallocation due to a missing reference to the expected `DBLQH` deallocation operation.

  We solve this issue by allowing commit of a scan operation in such cases, in order to release locks previously acquired during the transaction. We also add a new special case for this scenario, so that the deallocation is performed in a single phase, and `DBACC` tells `DBLQH` to deallocate immediately; in `DBLQH`, `execTUP_DEALLOCREQ()` is now able to handle this immediate deallocation request. (Bug #32491105)

  References: See also: Bug #28893633, Bug #32997832.

- Cluster nodes sometimes reported `Failed to convert connection to transporter` warnings in logs, even when this was not really necessary. (Bug #14784707)

- When started with no connection string on the command line, `ndb_waiter` printed `Connecting to mgmsrv at (null)`. Now in such cases, it prints `Connecting to management server at nodeid=0,localhost:1186` if no other default host is specified.

  The `--help` option and other `ndb_waiter` program output was also improved. (Bug #12380163)

- `NdbSpin_Init()` calculated the wrong number of loops in `NdbSpin`, and contained logic errors. (Bug #108448, Bug #32497174, Bug #32594825)

  References: See also: Bug #31765660, Bug #32413458, Bug #102506, Bug #32478388.

Changes in MySQL NDB Cluster 8.0.32 (2023-01-17, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Added the `--config-binary-file` option for `ndb_config`, which enables this program to read configuration information from the management server's binary configuration cache. This can be useful, for example, in determining whether or not the current version of the `config.ini` file has actually been read by the management server and applied to the cluster. See the description of the option in the MySQL NDB Cluster documentation, for more information and examples. (Bug #34773752)

Bugs Fixed

- **Packaging:** The man page for `ndbxfrm` was not present following installation. (Bug #34520046)
- **Solaris; NDB Client Programs:** `ndb_top` was not built for Solaris platforms. (Bug #34186837)
- **MySQL NDB ClusterJ:** ClusterJ could not be built on Ubuntu 22.10 with GCC 12.2. (Bug #34666985)
- In some contexts, a data node process may be sent `SIGCHLD` by other processes. Previously, the data node process bound a signal handler treating this signal as an error, which could cause the process to shut down unexpectedly when run in the foreground in a Kubernetes environment (and possibly under other conditions as well). This occurred despite the fact that a data node process never starts child processes itself, and thus there is no need to take action in such cases.

  To fix this, the handler has been modified to use `SIG_IGN`, which should result in cleanup of any child processes.

  Note

  `mysqld` and `ndb_mgmd` processes do not bind any handlers for `SIGCHLD`.

  (Bug #34826194)
- The running node from a node group scans each fragment (`CopyFrag`) and sends the rows to the starting peer in order to synchronize it. If a row from the fragment is locked exclusively by a user transaction, it blocks the scan from reading the fragment, causing the copyFrag to stall.

  If the starting node fails during the `CopyFrag` phase then normal node failure handling takes place. The cordinator node's transaction coordinator (TC) performs TC takeover of the user transactions from the TCs on the failed node. Since the scan that aids copying the fragment data over to the starting node is considered internal only, it is not a candidate for takeover, thus the takeover TC marks the `CopyFrag` scan as closed at the next opportunity, and waits until it is closed.

  The current issue arose when the `CopyFrag` scan was in the `waiting for row lock` state, and the closing of the marked scan was not performed. This led to TC takeover stalling while waiting for the close, causing unfinished node failure handling, and eventually a GCP stall potentially affecting redo logging, local checkpoints, and NDB Replication.

  We fix this by closing the marked `CopyFrag` scan whenever a node failure occurs while the `CopyFrag` is waiting for a row lock. (Bug #34823988)

  References: See also: Bug #35037327.
- In certain cases, invalid signal data was not handled correctly. (Bug #34787608)
- Sending of fragmented signals to virtual (`V_QUERY`) blocks is not supported, since the different signal fragments may end up in different block instances. When `DBTC` or `DBSPJ` sends a `LQHKEYREQ` or `SCAN_FRAGREQ` signal that may end up using `V_QUERY`, it checks whether the signal is fragmented and in that case changes the receiver to an instance of `DBLQH`. The function `SimulatedBlock::sendBatchedFragmentedSignal()` is intended to use the same check to decide whether to fragment a given signal, but did not, with the result that signals were fragmented which were not expected to be, sent using `V_QUERY`, and in that case likely to fail when received.

  We fix this problem by making the size check in `SimulatedBlock::sendFirstFragment()`, used by `sendBatchedFragmentedSignal()`, match the checks performed in `DBTC` and `DBSPJ`. (Bug #34776970)
- When the `DBSPJ` block submits `SCAN_FRAGREQ` requests to the local data managers, it usually scans only a subset of the fragments in parallel based on `recsPrKeys` statistics, if these are available, or just make a guess if no statistics are available.

SPJ contains logic which may take advantage of the result collected from the first round of fragments scanned; parallelism statistics are collected after SCAN_FRAGCONF replies are received, and first-match elimination may eliminate keys needed to scan in subsequent rounds.

Scanning local fragments is expected to have less overhead than scanning remote fragments, so it is preferable to err on the side of scan-parallelism for the local fragments. To take advantage of this, now two rounds are made over the fragments, the first one allowing SCAN_FRAGREQ signals to be sent to local fragments only, the second allowing such singals to be sent to any fragment expecting it. (Bug #34768216)

References: See also: Bug #34768191.

- When pushing a join to the data nodes, the query request is distributed to the SPJ blocks of all data nodes having local fragments for the first table (the SPJ root) in the pushed query. Each SPJ block retrieves qualifying rows from the local fragments of this root table, then uses the retrieved rows to generate a request to its joined child tables. If no qualifying rows are retrieved from the local fragments of the root, SPJ has no further work to perform.

  This implies that for a pushed join in which the root returns few rows, there are likely to be idling SPJ workers not taking full advantage of the available parallelism. Now for such queries we do not include very small tables in the pushed join, so that, if the next table in the join plan is larger, we start with that one instead. (Bug #34723413)

- The safety check for a copying ALTER TABLE operation uses the sum of per-fragment commit count values to determine whether any writes have been committed to a given table over a period of time. Different replicas of the same fragment do not necessarily have the same commit count over time, since a fragment replica's commit count is reset during node restart.

  Read primary tables always route read requests to a table's primary fragment replicas. Read backup and fully replicated tables optimize reads by allowing CommittedRead operations to be routed to backup fragment replicas. This results in the set of commit counts read not always being stable for Read backup and fully replicated tables, which can cause false positive failures for the copying ALTER TABLE safety check.

  This is solved by performing the copying ALTER TABLE safety check using a locking scan. Locked reads are routed to the same set of primary (main) fragments every time, which causes these counts to be stable. (Bug #34654470)

- Following execution of DROP NODEGROUP in the management client, attempting to creating or altering an NDB table specifying an explicit number of partitions or using MAX_ROWS was rejected with Got error 771 'Given NODEGROUP doesn't exist in this cluster' from NDB. (Bug #34649576)

- TYPE_NOTE_TRUNCATED and TYPE_NOTE_TIME_TRUNCATED were treated as errors instead of being ignored, as was the case prior to NDB 8.0.27. This stopped building of interpreted code for pushed conditions, with the condition being returned to the server.

  We fix this by reverting the handling of these status types to ignoring them, as was done previously. (Bug #34644930)

- When reorganizing a table with ALTER TABLE ... REORGANIZE PARTITION following addition of new data nodes to the cluster, fragments were not redistributed properly when the ClassicFragmentation configuration parameter was set to OFF. (Bug #34640773)

- Fixed an uninitialized padding variable in src/common/util/ndb_zlib.cpp. (Bug #34639073)

- When the NDB_STORED_USER privilege was granted to a user with an empty password, the user's password on each of the other SQL nodes was expired. (Bug #34626727)

- In a cluster with multiple management nodes, when one management node connected and later disconnected, any remaining management nodes were not aware of this node and were eventually forced to shut down when stopped nodes reconnected; this happened whenever the cluster still had live data nodes.

  On investigation it was found that node disconnection handling was done in the `NF_COMPLETEREP` path in `ConfigManager` but the expected `NF_COMPLETEREP` signal never actually arrived. We solve this by handling disconnecting management nodes when the `NODE_FAILREP` signal arrives, rather than waiting for `NF_COMPLETEREP`. (Bug #34582919)

- The `--diff-default` option and related options for `ndb_config` did not produce any usable output. (Bug #34549189)

  References: This issue is a regression of: Bug #32233543.

- Encrypted backups created on a system using one endian could not be restored on systems with the other endian; for example, encrypted backups taken on an x86 system could not be restored on a SPARC system, nor the reverse. (Bug #34446917)

- A query using a pushed join with an `IN` subquery did not return the expected result with `ndb_join_pushdown=ON` and the `BatchSize` SQL node parameter set to a very small value such as `1`. (Bug #34231718)

- When defining a binary log transaction, the transaction is kept in an in-memory binary log cache before it is flushed to the binary log file. If a binary log transaction exceeds the size of the cache, it is written to a temporary file which is set up early in the initialization of the binary log thread. This write introduces extra disk I/O in the binary log injector path. The number of disk writes performed globally by the binary log injector can be found by checking the value of the `Binlog_cache_disk_use` system status variable, but otherwise, the `NDB` handler's binary log injector thread had no way to observe this.

  Since `Binlog_cache_disk_use` is accessible by the binary log injector, it can be checked both before and after the transaction is committed to see whether there were any changes to its value. If any cache spills have taken place, this is reflected by the difference of the two values, and the binary log injector thread can report it. (Bug #33960014)

- When closing a file using compressed or encrypted format after reading the entire file, verify its checksum. (Bug #32550145)

- When reorganizing a table with `ALTER TABLE ... REORGANIZE PARTITION` following addition of new data nodes to the cluster, unique hash indexes were not redistributed properly. (Bug #30049013)

- For a partial local checkpoint, each fragment LCP must be to be able to determine the precise state of the fragment at the start of the LCP and the precise difference in the fragment between the start of the current LCP and the start of the previous one. This is tracked using row header information and page header information; in cases where physical pages are removed this is also tracked in logical page map information.

  A page included in the current LCP, before the LCP scan reaches it, is released due to the commit or rollback of some operation on the fragment, also releasing the last used storage on the page.

  Since the released page could not be found by the scan, the release itself set the `LCP_SCANNED_BIT` of the page map entry it was mapped into, in order to indicate that the page was already handled from the point of view of the current LCP, causing subsequent allocation and release of the pages mapped to the entry during the LCP to be ignored. The state of the entry at the start of the LCP was also set as allocated in the page map entry.

  These settings are cleared only when the next LCP is prepared. Any page release associated with the page map entry before the clearance would violate the requirement that the bit is not set; we resolve this issue by removing the (incorrect) requirement. (Bug #23539857)

- A data node could hit an overly strict assertion when the thread liveness watchdog triggered while the node was already shutting down. We fix the issue by relaxing this assertion in such cases. (Bug #22159697)
- Removed a leak of long message buffer memory that occurred each time an index was scanned for updating index statistics. (Bug #108043, Bug #34568135)
- `Backup::get_total_memory()`, used to calculate proposed disk write speeds for checkpoints, wrongly considered `DataMemory` that may not have been used in the calculation of memory used by LDMs.

  We fix this by obtaining the total `DataMemory` used by the LDM threads instead. as reported by `DBTUP`. (Bug #106907, Bug #34035805)
- Fixed an uninitialized variable in `Suma.cpp`. (Bug #106081, Bug #33764143)

Changes in MySQL NDB Cluster 8.0.31 (2022-10-11, General Availability)

- Transparent Data Encryption (TDE)
- RPM Notes
- Functionality Added or Changed
- Bugs Fixed

Transparent Data Encryption (TDE)

- This release implements Transparent Data Encryption (TDE), which provides protection by encryption of `NDB` data at rest. This includes all `NDB` table data and log files which are persisted to disk, and is intended to protect against recovering data subsequent to unauthorized access to NDB Cluster data files such as tablespace files or logs.

  To enforce encryption on files storing `NDB` table data, set `EncryptedFileSystem` to `1`, which causes all data to be encrypted and decrypted as necessary, as it is written to and read from these files. These include LCP data files, redo log files, tablespace files, and undo log files.

  When using file system encryption with `NDB`, you must also perform the following tasks:

  - Provide a password to each data node when starting or restarting it, using either one of the data node options `--filesystem-password` or `--filesystem-password-from-stdin`. This password uses the same format and is subject to the same constraints as the password used for an encrypted `NDB` backup (see the description of the `ndb_restore --backup-password` option).

    You can provide the encryption password on the command line, or in a `my.cnf` file. See NDB File System Encryption Setup and Usage, for more information and examples.

  Only tables using the `NDB` storage engine are subject to encryption by this feature; see NDB File System Encryption Limitations. Other tables, such as those used for `NDB` schema distribution, replication, and binary logging, typically use `InnoDB`; see InnoDB Data-at-Rest Encryption. For information about encryption of binary log files, see Encrypting Binary Log Files and Relay Log Files.

  Files generated or used by `NDB` processes, such as operating system logs, crash logs, and core dumps, are not encrypted. Files used by `NDB` but not containing any user table data are also not encrypted; these include LCP control files, schema files, and system files (see NDB Cluster Data Node File System). The management server configuration cache is also not encrypted.

  In addition, NDB 8.0.31 adds a new utility `ndb_secretsfile_reader` for extracting key information from encrypted files.

This enhancement builds on work done in NDB 8.0.22 to implement encrypted `NDB` backups. For more information, see the description of the `RequireEncryptedBackup` configuration parameter, as well as Using The NDB Cluster Management Client to Create a Backup.

> **Note**
>
> Upgrading an encrypted filesystem to NDB 8.0.31 or later from a previous release requires a rolling initial restart of the data nodes, due to improvements in key handling.

(Bug #34417282, WL #14687, WL #15051, WL #15204)

RPM Notes

- **ndbinfo Information Database:** Upgrades of SQL nodes from NDB 7.5 or NDB 7.6 to NDB 8.0 using RPM files did not enable the `ndbinfo` plugin properly. This was due to the fact that, since the `ndbcluster` plugin is disabled during an upgrade of `mysqld`, so is the `ndbinfo` plugin; this led to `.frm` files associated with `ndbinfo` tables being left behind following the upgrade.

  Now in such cases, any `ndbinfo` table `.frm` files from the earlier release are removed, and the plugin enabled. (Bug #34432446)

Functionality Added or Changed

- **Important Change; NDB Client Programs:** A number of NDB program options were never implemented and have now been removed. The options and the programs from which they have been dropped are listed here:

  - `--ndb-optimized-node-selection`: `ndbd`, `ndbmtd`, `ndb_mgm`, `ndb_delete_all`, `ndb_desc`, `ndb_drop_index`, `ndb_drop_table`, `ndb_show_tables`, `ndb_blob_tool`, `ndb_config`, `ndb_index_stat`, `ndb_move_data`, `ndbinfo_select_all`, `ndb_select_count`
  - `--character-sets-dir`: `ndb_mgm`, `ndb_mgmd`, `ndb_config`, `ndb_delete_all`, `ndb_desc`, `ndb_drop_index`, `ndb_drop_table`, `ndb_show_tables`, `ndb_blob_tool`, `ndb_config`, `ndb_index_stat`, `ndb_move_data`, `ndbinfo_select_all`, `ndb_select_count`, `ndb_waiter`
  - `--core-file`: `ndb_mgm`, `ndb_mgmd`, `ndb_config`, `ndb_delete_all`, `ndb_desc`, `ndb_drop_index`, `ndb_drop_table`, `ndb_show_tables`, `ndb_blob_tool`, `ndb_config`, `ndb_index_stat`, `ndb_move_data`, `ndbinfo_select_all`, `ndb_select_count`, `ndb_waiter`
  - `--connect-retries` and `--connect-retry-delay`: `ndb_mgmd`
  - `--ndb-nodeid`: `ndb_config`

  See the descriptions of the indicated program and program options in NDB Cluster Programs, for more information. (Bug #34059253)

- **Important Change:** The `ndbcluster` plugin is now included in all MySQL server builds, with the exception of builds for 32-bit platforms. As part of this work, we address a number of issues with `cmake` options for NDB Cluster, making the plugin option for `NDBCLUSTER` behave as other plugin options, and adding a new option `WITH_NDB` to control the build of NDB for MySQL Cluster.

  This release makes the following changes in `cmake` options relating to MySQL Cluster:

  - Adds the `WITH_NDB` option (default `OFF`). Enabling this option causes the MySQL Cluster binaries to be built.

- Deprecates the `WITH_NDBCLUSTER` option; use `WITH_NDB` instead.
- Removes the `WITH_PLUGIN_NDBCLUSTER` option. Use `WITH_NDB`, instead, to build MySQL Cluster.
- Changes the `WITH_NDBCLUSTER_STORAGE_ENGINE` option so that it now controls (only) whether the `ndbcluster` plugin itself is built. This option is now automatically set to `ON` when `WITH_NDB` is enabled for the build, so it should no longer be necessary to set it when compiling MySQL with NDB Cluster support.

For more information, see CMake Options for Compiling NDB Cluster. (WL #14788, WL #15157)

- Added the `--detailed-info` option for `ndbxfrm`. This is similar to the `--info` option, but in addition prints out the file's header and trailer. (Bug #34380739)
- This release makes it possible to enable and disable binary logging with compressed transactions using `ZSTD` compression for `NDB` tables in a `mysql` or other client session while the MySQL server is running. To enable the feature, set the `ndb_log_transaction_compression` system variable introduced in this release to `ON`. The level of compression used can be controlled using the `ndb_log_transaction_compression_level_zstd` system variable, which is also added in this release; the default compression level is `3`.

  Note

  Although changing the values of the `binlog_transaction_compression` and `binlog_transaction_compression_level_zstd` system variables from a client session has no effect on binary logging of `NDB` tables, setting `--binlog-transaction-compression=ON` on the command line or in a `my.cnf` file causes `ndb_log_transaction_compression` to be enabled, regardless of any setting for `--ndb-log-transaction-compression`. In this case, to disable binary log transaction compression for (only) `NDB` tables, set `ndb_log_transaction_compression=OFF` in a MySQL client session following startup of `mysqld`.

  For more information, see Binary Log Transaction Compression. (Bug #32704705, Bug #32927582, WL #15138, WL #15139)

Bugs Fixed

- When pushing a condition as part of a pushed join, it is a requirement that all *`table`*.*`column`* references are to one of the following:
  - The table to which the condition itself is pushed
  - A table which is an ancestor of the root of the pushed join
  - A table which is an ancestor of the table in the pushed query tree

  In the last case, when finding possible ancestors, we did not fully identify all candidates for such tables, in either or both of these two ways:

  1. Any tables being required ancestors due to nest-level dependencies were not added as ancestors
  2. Tables having all possible ancestors as either required ancestors or key parents are known to be directly joined with our ancestor, and to provide these as ancestors themselves; thus, such tables should be made available as ancestors as well.

This patch implements both cases 1 and 2. In the second case, we take a conservative approach and add only those tables having a `single row lookup` access type, but not those using index scans. (Bug #34508948)

- Execution of and `EXPLAIN` for some large join queries with ndb_join_pushdown enabled (the default) were rejected with NDB error `QRY_NEST_NOT_SUPPORTED` `FirstInner/Upper has to be an ancestor or a sibling`. (Bug #34486874)

- When the `NDB` join pushdown handler finds a table which cannot be pushed down it tries to produce an explanatory message communicating the reason for the rejection, which includes the names of the tables involved. In some cases the optimizer had already optimized away the table which meant that it could no longer be accessed by the `NDB` handler, resulting in failure of the query.

  We fix this by introducing a check for such cases and printing a more generic message which does not include the table name if no table is found. (Bug #34482783)

- The `EncryptedFilesystem` parameter was not defined with `CI_RESTART_INITIAL`, and so was not shown in the output of `ndb_config` as requiring `--initial`, even though the parameter does in fact require an initial restart to take effect. (Bug #34456384)

- When finding tables possible to push down in a pushed join, the pushability of a table may depend on whether later tables are pushed as well. In such cases we take an optimistic approach and assume that later tables are also pushed. If this assumption fails, we might need to “unpush” a table and any other tables depending on it. Such a cascading “unpush” may be due to either or both of the following conditions:

  - A key reference referred to a column from a table which later turned out to not be pushable.

  - A pushed condition referred to a column from a table which later turn out to not be pushable.

  We previously handled the first case, but handling of the second was omitted from work done in NDB 8.0.27 to enable pushing of conditions referring to columns from other tables that were part of the same pushed join. (Bug #34379950)

- `NdbScanOperation` errors are returned asynchronously to the client, possibly while the client is engaged in other processing. A successful call to `NdbTransaction::execute()` guarantees only that the scan request has been assembled and sent to the transaction coordinator without any errors; it does not wait for any sort of `CONF` or `REF` signal to be returned from the data nodes. In this particular case, the expected `TAB_SCANREF` signal was returned asynchronously into the client space, possibly while the client was still performing other operations.

  We make this behavior more deterministic by not setting the `NdbTransaction` error code when a `TAB_SCANREF` error is received. (Bug #34348706)

- When attempting to update a `VARCHAR` column that was part of an `NDB` table's primary key, the length of the value read from the database supplied to the `cmp_attr()` method was reportedly incorrectly. In addition to fixing this issue, we also remove an incorrect length check which required the binary byte length of the arguments to this method to be the same, which is not true of attributes being compared as characters, whose comparison semantics are defined by their character sets and collations. (Bug #34312769)

- When compiling NDB Cluster on OEL7 and OEL8 using `-Og` for debug builds, `gcc` raised a null pointer subtraction error. (Bug #34199675, Bug #34199732)

  References: See also: Bug #33855533.

- `ndb_blob_tool` did not perform proper handling of errors raised while reading data. (Bug #34194708)

- As part of setting up the signal execution strategy, we calculate a safe quota for the maximum numbers signals to execute from each job buffer. As each executed signal is assumed to generate

up to four outward bound signals, we might need to limit the signal quota so that we do not overfill the out buffers. Effectively, in each round of signal execution we cannot execute more signals than 1/4 of the signals that can fit in the out buffers.

This calculation did not take into account work done in NDB 8.0.23 introducing the possibility of having multiple writers, all using the same available free space in the same job buffer. Thus the signal quota needed to be further divided among the workers writing to the same buffers.

Now the computation of the maximum numbers signals to execute takes into account the resulting possibly greater number of writers to each queue. (Bug #34065930)

- When the `NDB` scheduler detects that job buffers are full, and starts to allocate from reserved buffers, it is expected to yield the CPU while waiting for the consumer. Just before yielding, it performs a final check for this condition, before sleeping. Problems arose when this check indicated that the job buffers were not full, so that the scheduler was allowed to continue executing signals, even though the limit on how many signals it was permitted to execute was still `0`. This led to a round of executing no signals, followed by another yield check, and so on, keeping the CPU occupied for no reason while waiting for something to be consumed by the receiver threads.

  The root cause of the problem was that different metrics were employed for calculating the limit on signals to execute (which triggered the yield check when this limit was `0`), and for the yield callback which subsequently checked whether the job buffers were actually full.

  Prior to the implementation of scalable job buffers in MySQL NDB Cluster 8.0.23, `NDB` waited for more job buffer up to 10 times; this was inadvertently changed so that it gave up after waiting one time only, despite log messages indicating that `NDB` had slept ten times. As part of this fix, we revert that change, so that, as before, we wait up to ten times for more job buffer before giving up. As an additional part of this work, we also remove extra (and unneeded) code previously added to detect spin waits. (Bug #34038016)

  References: See also: Bug #33869715, Bug #34025532.

- Job buffers act as the communication links between data node internal block threads. When the data structures for these were initialized, a 32K page was allocated to each such link, even if these threads never (by design) communicate with each other. This wasted memory resources, and had a small performance impact since the job buffer pages were checked frequently for available signals, so that us was necessary to load the unused job buffer pages into the translation lookaside buffer and L1, L2, and L3 caches.

  Now, instead, we set up an empty job buffer as a sentinel to which all the communication links refer initially. Actual (used) job buffer pages are now allocated only when we actually write signals into them, in the same way that new memory pages are allocated when a page gets full. (Bug #34032102)

- A data node could be forced to shut down due to a full job buffer, even when the local buffer was still available. (Bug #34028364)

- Made checks of pending signals by the job scheduler more consistent and reliable. (Bug #34025532)

  References: See also: Bug #33869715, Bug #34038016.

- The combination of batching with multiple in-flight operations per key, use of `IgnoreError`, and transient errors occurring on non-primary replicas led in some cases to inconsistencies within `DBTUP` resulting in replica misalignment and other issues. We now prevent this from happening by detecting when operations are failing on non-primary replicas, and forcing `AbortOnError` handling (rollback) in such cases for the containing transaction. (Bug #34013385)

- Handling by `ndb_restore` of temporary errors raised by DDL operations has been improved and made consistent. In all such cases, `ndb_restore` now retries the operation up to `MAX_RETRIES` (11) times before giving up. (Bug #33982499)

- Removed the causes of many warnings raised when compiling NDB Cluster. (Bug #33797357, Bug #33881953)
- When the rate of changes was high, event subscribers were slow to acknowledge receipt, or both, it was possible for the `SUMA` block to run out of space for buffering events. (Bug #30467140)
- `ALTER TABLE ... COMMENT="NDB_TABLE=READ_BACKUP=1"` or `ALTER TABLE..COMMENT="NDB_TABLE=READ_BACKUP=0"` performs a non-copying (online) `ALTER` operation on a table to add or remove its `READ_BACKUP` property (see NDB_TABLE Options), which increments the index version of all indexes on the table. Existing statistics, stored using the previous index version, were orphaned and never deleted; this led to wasted memory and inefficient searches when collecting index statistics.

  We address these issues by cleaning up the index samples; we delete any samples whose sample version is greater than or less than the current sample version. In addition, when no existing statistics are found by index ID and version, and when indexes are dropped. In this last case, we relax the bounds for the delete operation and remove all entries corresponding to the index ID in question, as opposed to both index ID and index version.

  This fix cleans up the sample table which stores the bulk of index statistics data. The head table, which consists of index metadata rather than actual statistics, still contains orphaned rows, but since these occupy an insignificant amount of memory, they do not adversely affect statistics search efficiency, and stale entries are cleaned up when index IDs and versions are reused.

  See also NDB API Statistics Counters and Variables. (Bug #29611297)

Changes in MySQL NDB Cluster 8.0.30 (2022-07-26, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change; NDB Replication:** The `replica_allow_batching` system variable affects how efficiently the replica applies epoch transactions. When this is set to `OFF`, by default, every discrete replication event in the binary log is applied and executed separately, which generally leads to poor performance.

  Beginning with this release, the default value for `replica_allow_batching` is changed from `OFF` to `ON`.
- **NDB Cluster APIs:** Removed a number of potential memory leaks by using `std::uniqe_ptr` for managing any `Event` returned by `Dictionary::getEvent()`.

  As part of this fix, we add a `releaseEvent()` method to `Dictionary` to clean up events created with `getEvent()` after they are no longer needed. (Bug #33855045)
- **NDB Cluster APIs:** The `Node.JS` package included with NDB Cluster has been updated to version 16.5.0. (Bug #33770627)
- Empty lines in CSV files are now accepted as valid input by `ndb_import`. (Previously, empty lines in such files were always rejected.) Now, if an empty value can be used as the value for a single imported column, `ndb_import` uses it in the same manner as `LOAD DATA`. (Bug #34119833)
- `NDB` stores blob column values differently from other types; by default, only the first 256 bytes of the value are stored in the table (“inline”), with any remainder kept in a separate blob parts table. This is true for columns of MySQL type `BLOB`, `MEDIUMBLOB`, `LONGBLOB`, `TEXT`, `MEDIUMTEXT`, and `LONGTEXT`. (`TINYBLOB` and `TINYTEXT` are exceptions, since they are always inline-only.) `NDB` handles `JSON` column values in a similar fashion, the only difference being that, for a `JSON` column, the first 4000 bytes of the value are stored inline.

Previously, it was possible to control the inline size for blob columns of NDB tables only by using the NDB API (Column::setInlineSize() method). This now can be done in the mysql client (or other application supplying an SQL interface) using a column comment which consists of an NDB_COLUMN string containing a BLOB_INLINE_SIZE specification, as part of a CREATE TABLE statement like this one:

```
CREATE TABLE t (
    a BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    b BLOB COMMENT 'NDB_COLUMN=BLOB_INLINE_SIZE=3000'
) ENGINE NDBCLUSTER;
```

In table t created by the statement just shown, column b (emphasized text in the preceding example) is a BLOB column whose first 3000 bytes are stored in t itself, rather than just the first 256 bytes. This means that, if no value stored in b exceeds 3000 bytes in length, no extra work is required to read or write any excess data from the NDB blob parts table when storing or retrieving the column value. This can improve performance significantly when performing many operations on blob columns.

You can see the effects of this option by querying the ndbinfo.blobs table, or examining the output of ndb_desc.

The maximum supported value for BLOB_INLINE_SIZE is 29980. Setting it to any value less than 1 causes the default inline size to be used for the column.

You can also alter a column as part of a copying ALTER TABLE; ALGORITHM=INPLACE is not supported for such operations.

BLOB_INLINE_SIZE can be used alone, or together with MAX_BLOB_PART_SIZE in the same NDB_COMMENT string. Unlike the case with MAX_BLOB_PART_SIZE, setting BLOB_INLINE_SIZE is supported for JSON columns of NDB tables.

For more information, see NDB_COLUMN Options, as well as String Type Storage Requirements. (Bug #33755627, WL #15044)

- A new --missing-ai-column option is added to ndb_import. This enables ndb_import to accept a CSV file from which the data for an AUTO_INCREMENT column is missing and to supply these values itself, much as LOAD DATA does. This can be done with one or more tables for which the CSV representation contains no values for such a column.

  This option works only when the CSV file contains no nonempty values for the AUTO_INCREMENT column to be imported. (Bug #102730, Bug #32553029)

- This release adds Performance Schema instrumentation for transaction batch memory used by NDBCLUSTER, making it possible to monitor memory used by transactions. For more information, see Transaction Memory Usage. (WL #15073)

Bugs Fixed

- **Important Change:** When using the ThreadConfig multithreaded data node parameter to specify the threads to be created on the data nodes, the receive thread (recv) in some cases was placed in the same worker thread as block threads such as DBLQH(0) and DBTC(0). This represented a regression from NDB 8.0.22 and earlier, where the receive thread is colocated only with THRMAN and TRPMAN, as expected in such cases.

  Now, when setting the value of ThreadConfig, you must include main, rep, recv, and ldm explicitly; to avoid using one or more of the main, rep, or ldm thread types, you must set count=0 explicitly for each applicable thread type.

  In addition, a minimum value of 1 is now enforced for the recv count; setting the replication thread (rep) count to 1 also requires setting count=1 for the main thread.

These changes can have serious implications for upgrades from previous NDB Cluster releases. For more information, see Upgrading and Downgrading NDB Cluster, as well as the description of the `ThreadConfig` parameter, in the MySQL Manual. (Bug #33869715)

References: See also: Bug #34038016, Bug #34025532.

- **macOS:** `ndb_import` could not be compiled on MacOS/ARM because the `ndbgeneral` library was not explicitly included in `LINK_LIBRARIES`. (Bug #33931512)

- **NDB Disk Data:** The `LGMAN` kernel block did not initialize its local encrypted filesystem state, and did not check `EncryptedFileSystem` for undo log files, so that their encryption status was never actually set.

  This meant that, for release builds, it was possible for the undo log files to be encrypted on some systems, even though they should not have been; in debug builds, undo log files were always encrypted. This could lead to problems when using Disk Data tables and upgrading to or from NDB 8.0.29. (A workaround is to perform initial restarts of the data nodes when doing so.)

  This issue could also cause unexpected CPU load for I/O threads when there were a great many Disk Data updates to write to the undo log, or at data node startup while reading the undo log.

  Note

  The `EncryptedFileSystem` parameter, introduced in NDB 8.0.29, is considered experimental and is not supported in production.

  (Bug #34185524)

- **NDB Cluster APIs:** The internal function `NdbThread_SetThreadPrio()` sets the thread priority (`thread_prio`) for a given thread type when applying the setting of the `ThreadConfig` configuration parameter. It was possible for this function in some cases to return an error when it had actually succeeded, which could have a an unfavorable impact on the performance of some NDB API applications. (Bug #34038630)

- **NDB Cluster APIs:** The following `NdbInterpretedCode` methods did not function correctly when a nonzero value was employed for the *`label`* argument:

  - `branch_col_and_mask_eq_mask()`
  - `branch_col_and_mask_eq_zero()`
  - `branch_col_and_mask_ne_mask()`
  - `branch_col_and_mask_ne_zero()`

  (Bug #33888962)

- **MySQL NDB ClusterJ:** ClusterJ support for systems with ARM-based Apple silicon is now enabled by default. (Bug #34148474)

- Compilation of NDB Cluster on Debian 11 and Ubuntu 22.04 halted during the link time optimization phase due to source code warnings being treated as errors. (Bug #34252425)

- `NDB` does not support in-place changes of default values for columns; such changes can be made only by using a copying `ALTER TABLE`. Changing of the default value in such cases was already detected, but the additional or removal of default value was not.

  We fix this issue by detecting when default value is added or removed during `ALTER TABLE`, and refusing to perform the operation in place. (Bug #34224193)

- After creating a user on SQL node A and granting it the `NDB_STORED_USER` privilege, dropping this user from SQL node B led to inconsistent results. In some cases, the drop was not distributed, so that after the drop the user still existed on SQL node A.

  The cause of this issue is that `NDB` maintains a cache of all local users with `NDB_STORED_USER`, but when a user was created on SQL node B, this cache was not updated. Later, when executing `DROP USER`, this led SQL node B to determine that the drop did not have to be distributed. We fix this by ensuring that this cache is updated whenever a new distributed user is created. (Bug #34183149)

- When the internal `ndbd_exit()` function was invoked on a data node, information and error messages sent to the event logger just prior to the `ndbd_exit()` call were not printed in the log as expected. (Bug #34148712)

- NDB Cluster did not compile correctly on Ubuntu 22.04 due to changes in OpenSSL 3.0. (Bug #34109171)

- NDB Cluster would not compile correctly using GCC 8.4 due to a change in Bison fallthrough handling. (Bug #34098818)

- Compiling the `ndbcluster` plugin or the `libndbclient` library required a number of files kept under directories specific to data nodes (`src/kernel`) and management servers (`src/mgmsrv`). These have now been moved to more suitable locations. Files moved that may be of interest are listed here:

  - `ndbd_exit_codes.cpp` is moved to `storage/ndb/src/mgmapi`
  - `ConfigInfo.cpp` is moved to `storage/ndb/src/common/mgmcommon`
  - `mt_thr_config.cpp` is moved to `storage/ndb/src/common`
  - `NdbinfoTables.cpp` is moved to `storage/ndb/src/common/debugger`

  (Bug #34045289)

- When an error occurred during the begin schema transaction phase, an attempt to update the index statistics automatically was made without releasing the transaction handle, leading to a leak. (Bug #34007422)

  References: See also: Bug #34992370.

- Path lengths were not always calculated correctly by the data nodes. (Bug #33993607)

- When `ndb_restore` performed an NDB API operation with any concurrent NDB API events taking place, contention could occur in the event of limited resources in `DBUTIL`. This led to temporary errors in `NDB`. In such cases, `ndb_restore` now attempts to retry the NDB API operation which failed. (Bug #33984717)

  References: See also: Bug #33982499.

- Removed a duplicate check of a table pointer found in the internal method `Dbtc::execSCAN_TABREQ()`. (Bug #33945967)

- The internal function `NdbReceiver::unpackRecAttr()`, which unpacks attribute values from a buffer from a `GSN_TRANSID_AI` signal, did not check to ensure that attribute sizes fit within the buffer. This could corrupt the buffer which could in turn lead to reading beyond the buffer and copying beyond destination buffers. (Bug #33941167)

- Improved formatting of log messages such that the format string verification employed by some compilers is no longer bypassed. (Bug #33930738)

- Some `NDB` internal signals were not always checked properly. (Bug #33896428)

- Fixed a number of issues in the source that raised `-Wunused-parameter` warnings when compiling NDB Cluster with GCC 11.2. (Bug #33881953)

- When an SQL node was not yet connected to `NDBCLUSTER`, an excessive number of warnings were written to the MySQL error log when the SQL node could not discover an `NDB` table. (Bug #33875273)

- The NDB API statistics variables `Ndb_api_wait_nanos_count`, `Ndb_api_wait_nanos_count_replica`, and `Ndb_api_wait_nanos_count_session` are used for determining CPU times and wait times for applications. These counters are intended to show the time spent waiting for responses from data nodes, but they were not entirely accurate because time spent waiting for key requests was not included.

  For more information, see NDB API Statistics Counters and Variables. (Bug #33840016)

  References: See also: Bug #33850590.

- It was possible in some cases for a duplicate `engine-se_private_id` entry to be installed in the MySQL data dictionary for an `NDB` table, even when the previous table definition should have been dropped.

  When data nodes drop out of the cluster and need to rejoin, each SQL node starts to synchronize the schema definitions in its own data dictionary. The `se_private_id` for an `NDB` table installed in the data dictionary is the same as its `NDB` table ID. It is common for tables to be updated with different IDs, such as when executing an `ALTER TABLE`, `DROP TABLE`, or `CREATE TABLE` statement. The previous table definition, obtained by referencing the table in *`schema.table`* format, is usually sufficient for a drop and thus for the new table to be installed with a new ID, since it is assumed that no other installed table definition uses that ID. An exception to this could occur during synchronization, if a data node shutdown allowed the previous table definition of a table having the same ID other than the one to be installed to remain, then the old definition was not dropped.

  To correct this issue, we now check whether the ID of the table to be installed in the data dictionary differs from that of the previous one, in which case we also check whether an old table definition already exists with that ID, and, if it does, we drop the old table before continuing. (Bug #33824058)

- After receiving a `COPY_FRAGREQ` signal, `DBLQH` sometimes places the signal in a queue by copying the signal object into a stored object. Problems could arise when this signal object was used to send another signal before the incoming `COPY_FRAGREQ` was stored; this led to saving a corrupt signal that, when sent, prevented a system restart from completing. We fix this by using a static copy of the signal for storage and retrieval, rather than the original signal object. (Bug #33581144)

- When the `mysqld` binary supplied with NDB Cluster was run without `NDB` support enabled, the `ndbinfo` and `ndb_transid_mysql_connection_map` plugins were still enabled, and for example, still shown with status `ACTIVE` in the output of `SHOW PLUGINS`. (Bug #33473346)

- Attempting to seize a redo log page could in theory fail due to a wrong bound error. (Bug #32959887)

- When a data node was started using the `--foreground` option, and with a node ID not configured to connect from a valid host, the data node underwent a forced shutdown instead of reporting an error. (Bug #106962, Bug #34052740)

- NDB tables were skipped in the MySQL Server upgrade phase and were instead migrated by the ndbcluster plugin at a later stage. As a result, triggers associated with NDB tables were not created during upgrades from 5.7 based versions.

  This occurred because it is not possible to create such triggers when the NDB tables are migrated by the ndbcluster plugin, since metadata about the triggers is lost in the upgrade finalization phase of the MySQL Server upgrade in which all .TRG files are deleted.

  To fix this issue, we make the following changes:

  - Migration of NDB tables with triggers is no longer deferred during the Server upgrade phase.
  - NDB tables with triggers are no longer removed from the data dictionary during setup even when initial system starts are detected.

  (Bug #106883, Bug #34058984)

- When initializing a file, NDBFS enabled autosync but never called do_sync_after_write() (then called sync_on_write()), so that the file was never synchronized to disk until it was saved. This meant that, for a system whose network disk was stalled for some time, the file could use up system memory on buffered file data.

  We fix this by calling do_sync_after_write() each time NDBFS writes to a file.

  As part of this work, we increase the autosync size from 1 MB to 16 MB when initializing files.

  Note

  NDB supports O_SYNC on platforms that provide it, but does not implement OM_SYNC for opening files.

  (Bug #106697, Bug #33946801, Bug #34131456)

Changes in MySQL NDB Cluster 8.0.29 (2022-04-26, General Availability)

Important

This release is no longer available for download. It was removed due to a critical issue that could cause data in InnoDB tables having added columns to be interpreted incorrectly. Please upgrade to MySQL Cluster 8.0.30 instead.

- Compilation Notes
- ndbinfo Information Database
- Performance Schema Notes
- Functionality Added or Changed
- Bugs Fixed

Compilation Notes

- NDB could not be built using GCC 11 due to an array out of bounds error. (Bug #33459671)
- Removed a number of -Wstringop-truncation warnings raised when compiling NDB with GCC 9 as well as suppression of such warnings. Also removed unneeded includes from the header file ndb_global.h. (Bug #32233543)

ndbinfo Information Database

- Eight new tables providing NDB dictionary information about database objects have been added to the ndbinfo information database. This makes it possible to obtain a great deal of information of this type by issuing queries in the mysql client, without the need to use ndb_desc, ndb_select_all, and similar utilities. (It is still be necessary to use ndb_desc to obtain fragment distribution information.) These tables are listed here, together with the NDB objects about which they provide information:

  - blobs: Blob tables
  - dictionary_columns: Table columns
  - dictionary_tables: Tables
  - events: Event subscriptions
  - files: Files used by disk data tables
  - foreign_keys: Foreign keys
  - hash_maps: Hash maps
  - index_columns: Table indexes

  An additional change in ndbinfo is that only files and hash_maps are defined as views; the remaining six tables listed previously are in fact base tables, even though they are not named using the ndb$ prefix. As a result, these tables are not hidden as other ndbinfo base tables are.

  For more information, see the descriptions of the tables in ndbinfo: The NDB Cluster Information Database. (WL #11968)

Performance Schema Notes

- ndbcluster plugin threads can now be seen in the Performance Schema. The threads and setup_threads tables show all three of these threads: the binary logging thread (ndb_binlog thread), the index statistics thread (ndb_index_stat thread), and the metadata thread (ndb_metadata thread).

  This makes it possible to obtain the thread IDs and thread OS IDs of these threads for use in queries on these and other Performance Schema tables.

  For more information and examples, see ndbcluster Plugin Threads. (WL #15000)

Functionality Added or Changed

- **NDB Cluster APIs:** The NDB API now implements a List::clear() method which clears all data from a list. This makes it simpler to reuse an existing list with the Dictionary methods listEvents(), listIndexes(), and listObjects().

  In addition, the List destructor has been modified such that it now calls clear() before attempting the removal of any elements or attributes from the list being destroyed. (Bug #33676070)

- The client receive thread was enabled only when under high load, where the criterion for determining “high load” was that the number of clients waiting in the poll queue (the receive queue) was greater than min_active_clients_recv_thread (default: 8).

  This was a poor metric for determining high load, since a single client, such as the binary log injector thread handling incoming replication events, could experience high load on its own as well. The same was true of a pushed join query (in which very large batches of incoming TRANSID_AI signals are received).

We change the receive thread such that it now sleeps in the poll queue rather than being deactivated completely, so that it is now always available for handling incoming signals, even when the client is not under high load. (Bug #33752914)

- It is now possible to restore the `ndb_apply_status` table from an `NDB` backup, using `ndb_restore` with the `--with-apply-status` option added in this release. In some cases, this information can be useful in new setting up new replication links.

  `--with-apply-status` restores all rows of the `ndb_apply_status` table except for the row for which the `server_id` value is `0`; use `--restore-epoch` to restore this row.

  To use the `--with-apply-status` option, you must also supply `--restore-data` when invoking `ndb_restore`.

  For more information, see the description of the `--with-apply-status` option in the Reference Manual, as well as ndb_apply_status Table. (Bug #32604161, Bug #33594652)

- Previously, when a user query attempted to open an `NDB` table with a missing (or broken) index, the MySQL server raised `NDB` error `4243 Index not found`. Now when such an attempt is made, it is handled as described here:

  - If the query does not make use of the problematic index, the query succeeds with no errors or warnings.

  - If the query attempts to use the missing or broken index, the query is rejected with a warning from `NDB` (`Index idx is not available in NDB. Use "ALTER TABLE tbl ALTER INDEX idx INVISIBLE" to prevent MySQL from attempting to access it, or use "ndb_restore --rebuild-indexes" to rebuild it`), and an error (`ER_NOT_KEYFILE`).

  The rationale for this change is that constraint violations or missing data sometimes make it impossible to restore an index on an `NDB` table, in which case, running `ndb_restore` with `--disable-indexes` restores the data without the index. With this change, once the data is restored from backup, it is possible to use SQL to fix any corrupt data and rebuild the index. (Bug #28584066, WL #14867)

Bugs Fixed

- **Important Change:** The maximum value supported for the `--ndb-batch-size` server option has been increased from `31536000` to `2147483648` (2 GB). (Bug #21040523)

- **Performance:** When profiling multithreaded data nodes (`ndbmtd`) performing a transaction including a large number of inserts, it was found that more than 50% of CPU time was spent in the internal method `Dblqh::findTransaction()`. It was found that, when there were many operations belonging to uncommitted transactions in the hash list searched by this method, the hash buckets overfilled, the result being that an excessive number of CPU cycles were consumed searching through the hash buckets.

  To address this problem, we fix the number of hash buckets at 4095, and scale the size of a hash bucket relative to the maximum number of operations, so that only relatively few items should now be placed in the same bucket. (Bug #33803541)

  References: See also: Bug #33803487.

- **Performance:** When inserting a great many rows into an empty or small table in the same transaction, the rate at which rows were inserted quickly declined to less than 50% of the initial rate; subsequently, it was found that roughly 50% of all CPU time was spent in `Dbacc::getElement()`, and the root cause identified to be the timing of resizing the structures used for storing elements by `DBACC`, growing with the insertion of more rows in the same transaction, and shrinking following a commit.

We fix this issue by checking for a need to resize immediately following the insertion or deletion of an element. This also handles the subsequent rejection of an insert. (Bug #33803487)

References: See also: Bug #33803541.

- **Performance:** A considerable amount of time was being spent searching the event buffer data hash (using the internal method `EventBufData_hash::search()`), due to the following issues:
  - The number of buckets proved to be too low under high load, when the hash bucket list could become very large.
  - The hash buckets were implemented using a linked list. Traversing a long linked list can be highly inefficient.

  We fix these problems by using a vector (`std::vector`) rather than a linked list, and by making the array containing the set of hash buckets expandable. (Bug #33796754)

- **Performance:** The internal function `computeXorChecksum()` was implemented such that great care was taken to aid the compiler in generating optimal code, but it was found that it consumed excessive CPU resources, and did not perform as well as a simpler implementation. This function is now reimplemented with a loop summing up `XOR` results over an array, which appears to result in better optimization with both GCC and Clang compilers. (Bug #33757412)

- **Microsoft Windows:** The `CompressedLCP` data node configuration parameter had no effect on Windows platforms.

  Note

  When upgrading to this release, Windows users should verify the setting for `CompressedLCP`; if it was previously enabled, you may experience an increase in CPU usage by I/O threads following the upgrade, when under load, when restoring data as part of a node restart, or in both cases. If this behavior is not desired, disable `CompressedLCP`.

  (Bug #33727690)

- **Microsoft Windows:** The internal function `Win32AsyncFile::rmrfReq()` did not always check for both **`ERROR_FILE_NOT_FOUND`** and **`ERROR_PATH_NOT_FOUND`** when either condition was likely. (Bug #33727647)

- **Microsoft Windows:** Corrected several minor issues that occurred with file handling on Windows platforms. (Bug #33727629)

- **NDB Cluster APIs:** Hash key generation using the internal API method `NdbBlob::getBlobKeyHash()` ignored the most significant byte of the key. This unnecessarily caused uneven distribution in the NDB API blob hash list, resulting in a increased need for comparing key values, and thus more CPU usage. (Bug #33803583)

  References: See also: Bug #33783274.

- **NDB Cluster APIs:** Removed an unnecessary assertion that could be hit when iterating through the list returned by `Dictionary::listEvents()`. (Bug #33630835)

- Builds on Ubuntu 21.10 using GCC 11 stopped with `-Werror=maybe-uninitialized`. (Bug #33976268)

- In certain cases, `NDB` did not handle node IDs of data nodes correctly. (Bug #33916404)

- In some cases, `NDB` did not validate all node IDs of data nodes correctly. (Bug #33896409)

- In some cases, array indexes were not handled correctly. (Bug #33896389, Bug #33896399, Bug #33916134)

- In some cases, integers were not handled correctly. (Bug #33896356)

- As part of work done in NDB 8.0.23 to implement the `AutomaticThreadConfig` configuration parameter, the maximum numbers of LQH and TC threads supported by `ndbmtd` were raised from 129 each to 332 and 160, respectively. This adversely affected the performance of `execSEND_PACKED()` methods implemented by several NDB kernel blocks, which complete sending of packed signals when the scheduler is about to suspend execution of the current block thread. This was due to continuing simply to iterate over the arrays of such threads despite the arrays' increased size. We fix this by using a bitmask to track the thread states alongside the full arrays. (Bug #33856371)

- When operating on blob columns, `NDB` must add extra operations to read and write the blob head column and blob part rows. These operations are added to the tail of the transaction's operation list automatically when the transaction is executed.

  To insert a new operation prior to a given operation, it was necessary to traverse the operation list from the beginning until the desired operation was found, with a cost proportional to the length L of the list of preceding operations. This is approximately $L^2 / 2$, increasing as more operations are added to the list; when a large number of operations modifying blobs were defined in a batch, this traversal cost was paid for each operation. This had a noticeable impact on performance when reading and writing blobs.

  We fix this by using list splicing in `NdbTransaction::execute()` to eliminate unnecessary traversals of this sort when defining blob operations. (Bug #33797931)

- The block thread scheduler makes frequent calls to `update_sched_config()` to update its scheduling strategy. That involves checking the fill degree of the job buffer queues used to send signals between the nodes' internal block threads. When these queues are about to fill up, the thread scheduler assigns a smaller value to `max_signals` for the next round, in order to reduce the pressure on the job buffers. When the minimum free threshold has been reached, the scheduler yields the CPU while waiting for the consumer threads to free some job buffer slots.

  The fix in NDB 8.0.18 for a previous issue introduced a mechanism whereby the main thread was allowed to continue executing even when this lower threshold had been reached; in some cases the main thread consumed all job buffers, including those held in reserve, leading to an unplanned shutdown of the data node due to resource exhaustion. (Bug #33792362, Bug #33872577)

  References: This issue is a regression of: Bug #29887068.

- Setting up a cluster with one LDM thread and one query thread using the `ThreadConfig` parameter (for example, `ThreadConfig=ldm={cpubind=1},query={cpubind=2}`) led to unplanned shutdowns of data nodes.

  This was due to internal thread variables being assigned the wrong values when there were no main or request threads explicitly assigned. Now we make sure in such cases that these are assigned the thread number of the first receive thread, as expected. (Bug #33791270)

- `NdbEventBuffer` hash key generation for non-character data reused the same 256 hash keys; in addition, strings of zero length were ignored when calculating hash keys. (Bug #33783274)

- The collection of NDB API statistics based on the `EventBytesRecvdCount` event counter incurred excessive overhead. Now this counter is updated using a value which is aggregated as the event buffer is filled, rather than traversing all of the event buffer data in a separate function call.

  For more information, see NDB API Statistics Counters and Variables. (Bug #33778923)

- The internal method `THRConfig::reorganize_ldm_bindings()` behaved unexpectedly, in some cases changing thread bindings after `AutomaticThreadConfig` had already bound the threads to the correct CPUs. We fix this by removing the method, no longer using it when parsing configuration data or adding threads. (Bug #33764260)

- The receiver thread ID was hard-coded in the internal method `TransporterFacade::raise_thread_prio()` such that it always acted to raise the priority of the receiver thread, even when called from the send thread. (Bug #33752983)

- A fix in NDB 8.0.28 addressed an issue with the code used by various `NDB` components, including `Ndb_index_stat`, that checked whether the data nodes were up and running. In clusters with multiple SQL nodes, this resulted in an increase in the frequency of race conditions between index statistics threads trying to create a table event on the `ndb_index_stat_head` table; that is, it was possible for two SQL nodes to try to create the event at the same time, with the losing SQL node raising Error 746 `Event name already exists`. Due to this error, the binary logging thread ended up waiting for the index statistics thread to signal that its own setup was complete, and so the second SQL node timed out with `Could not create index stat system tables` after `--ndb-wait-setup` seconds. (Bug #33728909)

  References: This issue is a regression of: Bug #32019119.

- On a write error, the message printed by `ndbxfrm` referenced the source file rather than the destination file. (Bug #33727551)

- A complex nested join was rejected with the error `FirstInner/Upper has to be an ancestor or a sibling`, which is thrown by the internal `NdbQueryOperation` interface used to define a pushed join in the SPJ API, indicating that the join-nest dependencies for the interface were not properly defined.

  The query showing the issue had the join nest structure `t2, t1, (t3, (t5, t4))`. Neither of the join conditions on `t5` or `t4` had any references or explicit dependencies on table `t3`, but each had an implicit dependency on `t3` in virtue of being in a nest within the same nest as `t3`.

  When preparing a pushed join, `NDB` tracks all required table dependencies between tables and join-nests by adding them to the `m_ancestor` bitmask for each table. For nest level dependencies, they should all be added to the first table in the relevant nest. When the relevant dependencies for a specific table are calculated, they include the set of all tables being explicitly refered in the join condition, plus any implicit dependencies due to the join nests the table is a member of, limited by the uppermost table referred to in the join condition.

  For this particular join query we did not properly take into account that there might not be any references to tables in the closest upper nest (the nest starting with `t3`); in such cases we are dependent on all nests up to the nest containing the uppermost table referenced. We fix the issue by introducing a while-loop in which we add ancestor nest dependencies until we reach this uppermost table. (Bug #33670002)

- When the transient memory pool (`TransientPool`) used internally by `NDB` grew above 256 MB, subsequent attempts to shrink the pool caused an error which eventually led to an unplanned shutdown of the data node. (Bug #33647601)

- Check that the connection to `NDB` has been set up before querying about statistics for partitions. (Bug #33643512)

- When the ordered index `PRIMARY` was not created for the `ndb_sql_metadata` table, application of stored grants could not proceed due to the missing index.

  We fix this by protecting creation of utility tables (including ndb_sql_metadata) by wrapping the associated `CREATE TABLE` statement with a schema transaction, thus handling rejection of the statement by rollback. In addition, in the event the newly-created table is not created correctly, it is dropped. These changes avoid leaving behind a table that is only partially created, so that the next attempt to create the utility table starts from the beginning of the process. (Bug #33634453)

- Removed `-Wmaybe-uninitialized` warnings which occurred when compiling NDB Cluster with GCC 11.2. (Bug #33611915)

- NDB accepted an arbitrary (and invalid) string of characters following a numeric parameter value in the config.ini global configuration. For example, it was possible to use either OverloadLimit=10 "M12L" or OverloadLimit=10 M (which contains a space) and have it interpreted as OverloadLimit=10M.

  It was also possible to use a bare letter suffix in place of an expected numeric value, such as OverloadLimit=M, and have it interpreted as zero. This happened as well with an arbitrary string whose first letter was one of the MySQL standard modifiers K, M, or G; thus, OverloadLimit=MAX_UINT also had the effect of setting OverloadLimit to zero.

  Now, only one of the suffixes K, M, or G is accepted with a numeric parameter value, and it must follow the numeric value immediately, with no intervening whitespace characters or quotation marks. In other words, to set OverloadLimit to 10 megabytes, you must use one of OverloadLimit=10000000, OverloadLimit=10M, or OverloadLimit=10000K.

  Note

  To maintain availability, you should check your config.ini file for any settings that do not conform to the rule enforced as a result of this change and correct them prior to upgrading. Otherwise, the cluster may not be able to start afterwards, until you rectify the issue.

  (Bug #33589961)

- Enabling AutomaticThreadConfig with fewer than 8 CPUs available led to unplanned shutdowns of data nodes. (Bug #33588734)

- Removed the unused source files buddy.cpp and buddy.hpp from storage/ndb/src/common/transporter/. (Bug #33575155)

- The NDB stored grants mechanism now sets the session variable print_identified_with_as_hex to true, so that password hashes stored in the ndb_sql_metadata table are formatted as hexadecimal values rather than being formatted as strings. (Bug #33542052)

- Binary log thread event handling includes optional high-verbosity logging, which, when enabled and the connection to NDB lost, produces an excess of log messages like these:

  ```
  datetime 2 [Note] [MY-010866] [Server] NDB Binlog: cluster failure for epoch 55/0.
  datetime 2 [Note] [MY-010866] [Server] NDB Binlog: cluster failure for epoch 55/0.
  ```

  Such repeated log messages, not being of much help in diagnosing errors, have been removed. This leaves a similar log message in such cases, from the handling of schema distribution event operation teardown. (Bug #33492244)

- Historically, a number of different methods have been used to enforce compile-time checks of various interdependencies and assumptions in the NDB codebase in a portable way. Since the standard static_assert() function is now always available, the NDB_STATIC_ASSERT and STATIC_ASSERT macros have been replaced with direct usage of static_assert(). (Bug #33466577)

- When the internal AbstractQueryPlan interface determined the access type to be used for a specific table, it tried to work around an optimizer problem where the ref access type was specified for a table and later turned out to be accessible by eq_ref. The workaround introduced a new issue by sometimes determining eq_ref access for a table actually needing ref access; in addition, the prior fix did not take into account UNIQUE USING HASH indexes, which need either eq_ref or full table scan access, even when the MySQL Optimizer regards it as a ref access.

  We fix this by first removing the workaround (which had been made obsolete by the proper fix for the previous issue), and then by introducing the setting of eq_ref or full_table_scan access for hash indexes. (Bug #33451256)

References: This issue is a regression of: Bug #28965762.

- When a pushed join is prepared but not executed, the `Ndb_pushed_queries_dropped` status variable is incremented. Now, in addition to this, `NDB` now emits a warning `Prepared pushed join could not be executed...` which is passed to `ER_GET_ERRMSG`. (Bug #33449000)

- The deprecated `-r` option for `ndbd` has been removed. In addition, this change also removes extraneous text from the output of `ndbd --help`. (Bug #33362935)

  References: See also: Bug #31565810.

- `ndb_import` sometimes could not parse correctly a `.csv` file containing Windows/DOS-style (`\r\n`) linefeeds. (Bug #32006725)

- The `ndb_import` tool handled only the hidden primary key which is defined by `NDB` when a table does not have an explicit primary key. This caused an error when inserting a row containing `NULL` for an auto-increment primary key column, even though the same row was accepted by `LOAD DATA INFILE`.

  We fix this by adding support for importing a table with one or more instances of `NULL` in an auto-increment primary key column. This includes a check that a table has no more than one auto-increment column; if this column is nullable, it is redefined by `ndb_import` as `NOT NULL`, and any occurrence of `NULL` in this column is replaced by a generated auto-increment value before inserting the row into `NDB`. (Bug #30799495)

- When a node failure is detected, surviving nodes in the same nodegroup as this node attempt to resend any buffered change data to event subscribers. In cases in which there were no outstanding epoch deliveries, that is, the list of unacknowledged GCIs was empty, the surviving nodes made the incorrect assumption that this list would never be empty. (Bug #30509416)

- When executing a copying `ALTER TABLE` of the parent table for a foreign key and the SQL node terminates prior to completion, there remained an extraneous temporary table with (additional, temporary) foreign keys on all child tables. One consequence of this issue was that it was not possible to restore a backup made using `mysqldump --no-data`.

  To fix this, `NDB` now performs cleanup of temporary tables whenever a `mysqld` process connects (or reconnects) to the cluster. (Bug #24935788, Bug #29892252)

- An unplanned data node shutdown occurred following a bus error on Mac OS X for ARM. We fix this by moving the call to `NdbCondition_Signal()` (in `AsyncIoThread.cpp`) such that it executes prior to `NdbMutex_Unlock()`—that is, into the mutex, so that the condition being signalled is not lost during execution. (Bug #105522, Bug #33559219)

- In `DblqhMain.cpp`, a missing return in the internal `execSCAN_FRAGREQ()` function led to an unplanned shutdown of the data node when inserting a nonfatal error. In addition, the condition `!seize_op_rec(tcConnectptr)` present in the same function was never actually checked. (Bug #105051, Bug #33401830, Bug #33671869)

- It was possible to set any of `MaxNoOfFiredTriggers`, `MaxNoOfLocalScans`, and `MaxNoOfLocalOperations` concurrently with `TransactionMemory`, although this is not allowed.

  In addition, it was not possible to set any of `MaxNoOfConcurrentTransactions`, `MaxNoOfConcurrentOperations`, or `MaxNoOfConcurrentScans` concurrently with `TransactionMemory`, although there is no reason to prevent this.

  In both cases, the concurrent settings behavior now matches the documentation for the `TransactionMemory` parameter. (Bug #102509, Bug #32474988)

- When a redo log part is unable to accept an operation's log entry immediately, the operation (a prepare, commit, or abort) is queued, or (prepare only) optionally aborted. By default operations are queued.

This mechanism was modified in 8.0.23 as part of decoupling local data managers and redo log parts, and introduced a regression whereby it was possible for queued operations to remain in the queued state until all activity on the log part quiesced. When this occurred, operations could remain queued until `DBTC` declared them timed out, and aborted them. (Bug #102502, Bug #32478380)

Changes in MySQL NDB Cluster 8.0.28 (2022-01-18, General Availability)

- Compilation Notes
- Functionality Added or Changed
- Bugs Fixed

Compilation Notes

- `NDB` did not compile using GCC 11 on Ubuntu 21.10. (Bug #33424843)

Functionality Added or Changed

- Added the `ndbinfo index_stats` table, which provides very basic information about `NDB` index statistics. It is intended primarily for use in our internal testing, but may be helpful in conjunction with `ndb_index_stat` and other tools. (Bug #32906654)
- Previously, `ndb_import` always tried to import data into a table whose name was derived from the name of the CSV file being read. This release adds a `--table` option (short form: `-t`) for this program, which overrides this behavior and specifies the name of the target table directly. (Bug #30832382)

Bugs Fixed

- **Important Change:** The deprecated data node option `--connect-delay` has been removed. This option was a synonym for `--connect-retry-delay`, which was not honored in all cases; this issue has been fixed, and the option now works correctly. In addition, the short form `-r` for this option has been deprecated, and you should expect it to be removed in a future release. (Bug #31565810)

  References: See also: Bug #33362935.
- **NDB Cluster APIs:** Several new basic example C++ NDB API programs have been added to the distribution, under `storage/ndb/ndbapi-examples/ndbapi_basic/` in the source tree. These are shorter and should be easier to understand for newcomers to the NDB API than the existing API examples. They also follow recent C++ standards and practices. These examples have also been added to the NDB API documentation; see Basic NDB API Examples, for more information. (Bug #33378579, Bug #33517296)
- **NDB Cluster APIs:** It is no longer possible to use the `DIVERIFYREQ` signal asynchronously. (Bug #33161562)
- Timing of `wait for scans` log output during online reorganization was not performed correctly. As part of this fix, we change timing to generate one message every 10 seconds rather than scaling indefinitely, so as to supply regular updates. (Bug #35523977)
- Added missing values checks in `ndbd` and `ndbmtd`. (Bug #33661024)
- Online table reorganization increases the number of fragments of a table, and moves rows between them. This is done in the following steps:

  1. Copy rows to new fragments
  2. Update distribution information (hashmap count and total fragments)

3. Wait for scan activity using old distribution to stop
4. Delete rows which have moved out of existing partitions
5. Remove reference to old hashmap
6. Wait for scan activity started since step 2 to stop

Due to a counting error, it was possible for the reorganization to hang in step 6; the scan reference count was not decremented, and thus never reached zero as expected. (Bug #33523991)

- A `UNIQUE` index created with `USING HASH` does not support ordered or range access operations, but rather only those operations in which the full key is specified, returning at most a single row. Even so, for such an index on an NDB table, range access was still used on the index. (Bug #33466554, Bug #33474345)
- The same pushed join on `NDB` tables returned an incorrect result when the `batched_key_access` optimizer switch was enabled.

  This issue arose as follows: When the batch key access (BKA) algorithm is used to join two tables, a set of batched keys is first collected from one of the tables; a multirange read (MRR) operation is constructed against the other. A set of bounds (ranges) is specified on the MRR, using the batched keys to construct each bound.

  When result rows are returned it is necessary to identify which range each returned row comes from. This is used to identify the outer table row to perform the BKA join with. When the MRR operation in question was a root of a pushed join operation, `SPJ` was unable to retrieve this identifier (`RANGE_NO`). We fix this by implementing the missing `SPJ` API functionality for returning such a `RANGE_NO` from a pushed join query. (Bug #33416308)
- Each query against the `ndinfo.index_stats` table leaked an `NdbRecord`. We fix this by changing the context so that it owns the `NdbRecord` object which it creates and then to release the `NdbRecord` when going out of scope, and by supporting the creation of one and only one record per context. (Bug #33408123)
- A problem with concurrency occurred when updating cached table statistics with changed rows, when several threads updating same table the threads competed for the `NDB_SHARE` mutex in order to update the cached row count.

  We fix this by reimplementing the storage of changed rows using an atomic counter rather than trying to take the mutex and update the actual shared value, which reduces the need to serialize the threads. In addition, we now append the number of changed rows to the row count only when removing the statistics from the cache and provide a separate mutex protecting only the cached statistics. (Bug #33384978)

  References: See also: Bug #32169848.
- If the schema distribution client detected a timeout before freeing the schema object when the coordinator received the schema event, the coordinator processed the stale schema event instead of returning.

  The coordinator did not know whether a schema distribution timeout was detected by the client, and started processing the schema event as soon as the schema object was valid. To fix this, we indicate the state of the schema object and change its state when the client detects the schema distribution timeout and when the schema event is received by the coordinator, so that both the coordinator and the client are aware of this, and remain synchronized. (Bug #33318201)
- The MySQL Optimizer uses two different methods, `handler::read_cost()` and `Cost_model::page_read_cost()`, to estimate the cost for different access methods, but the cost values returned by these were not always comparable; in some cases this led to the wrong index

being chosen and longer execution time for effected queries. To fix this for NDB, we override the optimizer's page_read_cost() method with one specific to NDBCLUSTER. It was also found while working on this issue that the NDB handler did not implement the read_time() method, used by read_cost(); this method is now implemented by ha_ndbcluster, and thus the optimizer can now properly take into account the cost difference for NDB when using a unique key as opposed to an ordered index (range scan). (Bug #33317872)

- When opening NDB tables for queries, the index statistics are retrieved to help the optimizer select the optimal query plan. Each client accessing the stats acquires the global index statistics mutex both before and after accessing the statistics. This causes mutex contention affecting query performance, whether or not there are queries are operating on the same tables, or on different ones.

  We fix this by protecting the count of index statistics references with an atomic counter. The problem was clearly visible when benchmarking with more than 32 clients, when throughput did not increase with additional clients. With this fix, the throughput continues to scale with up to 64 clients. (Bug #33317320)

- In certain cases, an event's category was not properly detected. (Bug #33304814)

- It was not possible to add new data nodes running ndbd to an existing cluster with data nodes running ndbd. (Bug #33193393)

- For a user granted the NDB_STORED_USER privilege, the password_last_changed column in the mysql.user table was updated each time the SQL node was restarted. (Bug #33172887)

- DBDICT did not always perform table name checks correctly. (Bug #33161548)

- Added a number of missing ID and other values checks in ndbd and ndbmtd. (Bug #33161486, Bug #33162047)

- Added a number of missing ID and other values checks in ndbd and ndbmtd. (Bug #33161259, Bug #33161362)

- SET_LOGLEVELORD signals were not always handled correctly. (Bug #33161246)

- DUMP 11001 did not always handle all of its arguments correctly. (Bug #33157513)

- File names were not always verified correctly. (Bug #33157475)

- Added a number of missing checks in the data nodes. (Bug #32983723, Bug #33157488, Bug #33161451, Bug #33161477, Bug #33162082)

- Added a number of missing ID and other values checks in ndbd and ndbmtd. (Bug #32983700, Bug #32893708, Bug #32957478, Bug #32983256, Bug #32983339, Bug #32983489, Bug #32983517, Bug #33157527, Bug #33157531, Bug #33161271, Bug #33161298, Bug #33161314, Bug #33161331, Bug #33161372, Bug #33161462, Bug #33161511, Bug #33161519, Bug #33161537, Bug #33161570, Bug #33162059, Bug #33162065, Bug #33162074, Bug #33162082, Bug #33162092, Bug #33162098, Bug #33304819)

- The management server did not always handle events of the wrong size correctly. (Bug #32957547)

- When ndb_mgmd is started without the --config-file option, the user is expected to provide the connection string for another management server in the same cluster, so that the management server being started can obtain configuration information from the other. If the host address in the connection string could not be resolved, then the ndb_mgmd being started hung indefinitely while trying to establish a connection.

  This issue occurred because a failure to connect was treated as a temporary error, which led to the ndb_mgmd retrying the connection, which subsequently failed, and so on, repeatedly. We fix this by treating a failure in host name resolution by ndb_mgmd as a permanent error, and immediately exiting. (Bug #32901321)

- The order of parameters used in the argument to `ndb_import --csvopt` is now handled consistently, with the rightmost parameter always taking precedence. This also applies to duplicate instances of a parameter. (Bug #32822757)
- In some cases, issues with the redo log while restoring a backup led to an unplanned shutdown of the data node. To fix this, when the redo log file is not available for writes, we now include the correct wait code and waiting log part in the `CONTINUEB` signal before sending it. (Bug #32733659)

  References: See also: Bug #31585833.
- The binary logging thread sometimes attempted to start before all data nodes were ready, which led to excess logging of unnecessary warnings and errors. (Bug #32019919)
- Instituted a number of value checks in the internal `Ndb_table_guard::getTable()` method. This fixes a known issue in which an SQL node underwent an unplanned shutdown while executing `ALTER TABLE` on an `NDB` table, and potentially additional issues. (Bug #30232826)
- Replaced a misleading error message and otherwise improved the behavior of `ndb_mgmd` when the `HostName` could not be resolved. (Bug #28960182)
- A query used by MySQL Enterprise Monitor to monitor memory use in NDB Cluster became markedly less performant as the number of `NDB` tables increased. We fix this as follows:
  - Row counts for virtual `ndbinfo` tables have been made available to the MySQL optimizer
  - Size estimates are now provided for all `ndbinfo` tables
  - Primary keys have been added to most internal `ndbinfo` tables

  Following these improvements, the performance of queries against `ndbinfo` tables should be comparable to queries against equivalent `MyISAM` tables. (Bug #28658625)
- Following improvements in LDM performance made in NDB 8.0.23, an `UPDATE_FRAG_DIST_KEY_ORD` signal was never sent when needed to a data node using node ID `1`. When running the cluster with 3 or 4 replicas and another node in the same node group restarted, this could result in SQL statements being rejected with error MySQL 1297 `ER_GET_TEMPORARY_ERRMSG` and, subsequently, `SHOW WARNINGS` reporting error NDB error `1204`.

  Note

  Prior to upgrading to this release, you can work around the issue by restarting data node 1 whenever any other node in the same node group has been restarted.

  (Bug #105098, Bug #33460188)
- Following the rolling restart of a data node performed as part of an upgrade from NDB 7.6 to NDB 8.0, the data node underwent a forced shutdown. We fix this by allowing `LQHKEYREQ` signals to be sent to both the `DBLQH` and the `DBSPJ` kernel blocks. (Bug #105010, Bug #33387443)
- When the `AutomaticThreadConfig` parameter was enabled, `NumCPUs` was always shown as `0` in the data node log. In addition, when this parameter is in use, thread CPU bindings are now made correctly, and the data node log shows the actual CPU binding for each thread. (Bug #102503, Bug #32474961)
- `ndb_blob_tool --help` did not return the expected output. (Bug #98158, Bug #30733508)
- `NDB` did not close any pending schema transactions when returning an error from internal system table creation and drop functions.

Changes in MySQL NDB Cluster 8.0.27 (2021-10-19, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change:** The old `MaxAllocate` data node configuration parameter has no effect in any current version of `NDB`. As of this release, it is deprecated and subject to removal in a future release. (Bug #52980, Bug #11760559)

- **NDB Cluster APIs:** Conditions pushed as part of a pushed query can now refer to columns from ancestor tables within the same pushed query.

  For example, given a table created using `CREATE TABLE t (x INT PRIMARY KEY, y INT) ENGINE=NDB`, the query such as that shown here can now employ condition pushdown:

  ```
  SELECT * FROM t AS a
  LEFT JOIN t AS b
  ON a.x=0 AND b.y>5,
  ```

  Pushed conditions may include any of the common comparison operators `<`, `<=`, `>`, `>=`, `=`, and `<>`.

  Values being compared must be of the same type, including length, precision, and scale.

  `NULL` handling is performed according to the comparison semantics specified by the ISO SQL standard; any comparison with `NULL` returns `NULL`.

  For more information, see Engine Condition Pushdown Optimization.

  As part of this work, the following `NdbInterpretedCode` methods are implemented in the NDB API for comparing column values with values of parameters:

  - `branch_col_eq_param()`
  - `branch_col_ne_param()`
  - `branch_col_lt_param()`
  - `branch_col_le_param()`
  - `branch_col_gt_param()`
  - `branch_col_ge_param()`

  In addition, a new `NdbScanFilter::cmp_param()` API method makes it possible to define comparisons between column values and parameter values. (WL #14388)

- In environments that monitor and disconnect idle TCP connections, an idle cluster could suffer from unnecessary data node failures, and the failure of more than one data node could lead to an unplanned shutdown of the cluster.

  To fix this problem, we introduce a new keep-alive signal (`GSN_TRP_KEEP_ALIVE`) that is sent on all connections between data nodes on a regular basis, by default once every 6000 milliseconds (one minute). The length of the interval between these signals can be adjusted by setting the `KeepAliveSendInterval` data node parameter introduced in this release, which can be set to 0 to disable keep-alive signals. You should be aware that `NDB` performs no checking that these signals are received and performs no disconnects on their account (this remains the responsibility of the heartbeat protocol). (Bug #32776593)

- A copying `ALTER TABLE` now checks the source table's fragment commit counts before and after performing the copy. This allows the SQL node executing the `ALTER TABLE` to determine whether there has been any concurrent write activity to the table being altered, and, if so, to terminate the

operation, which can help avoid silent loss or corruption of data. When this occurs, the `ALTER TABLE` statement is now rejected with the error `Detected change to data in source table during copying ALTER TABLE. Alter aborted to avoid inconsistency`. (Bug #24511580, Bug #25694856, WL #10540)

- The creation and updating of `NDB` index statistics are now enabled by default. In addition, when restoring metadata, `ndb_restore` now creates the index statistics tables if they do not already exist. (WL #14355)

Bugs Fixed

- **Important Change; NDB Cluster APIs:** Since MySQL 8.0 uses the data dictionary to store table metadata, the following NDB API `Table` methods relating to `.FRM` files are now deprecated:
  - `getFrmData()`
  - `getFrmLength()`
  - `setFrm()`

  NDB 8.0 uses `getExtraMetadata()` and `setExtraMetadata()` for reading and writing table metadata stored in the MySQL data dictionary; you should expect the `*Frm*()` methods listed previously to be removed in a future release of NDB Cluster. (Bug #28248575)
- **Important Change:** The default value for each of the two `mysqld` options `--ndb-wait-connected` and `--ndb-wait-setup` has been increased from 30 to 120 seconds. (Bug #32850056)
- **Microsoft Windows:** A number of warnings generated when building NDB Cluster and the NDB utilities with Visual Studio 16.9.5 were removed. (Bug #32881961)
- **Microsoft Windows:** On Windows, it was not possible to start data nodes successfully when the cluster was configured to use more than 64 data nodes. (Bug #104682, Bug #33262452)
- **NDB Cluster APIs:** A number of MGM API functions, including `ndb_mgm_create_logevent_handle()`, did not release memory properly. (Bug #32751506)
- **NDB Cluster APIs:** Trying to create an index using `NdbDictionary` with index statistics enabled and the index statistics tables missing resulted in `NDB` error 723 `No such table existed`, the missing table in this context being one of the statistics tables, which was not readily apparent to the user. Now in such cases, `NDB` instead returns error 4714 `Index stats system tables do not exist`, which is added in this release. (Bug #32739080)
- **NDB Cluster APIs:** The MySQL NoSQL Connector for JavaScript included with NDB Cluster is now built using Node.js version 12.2.6.
- A buffer used in the `SUMA` kernel block did not always accommodate multiple signals. (Bug #33246047)
- In `DbtupBuffer.cpp` the priority level is adjusted to what is currently executing in one path, but it was not used for short signals. This leads to the risk of `TRANSID_AI` signals, `SCAN_FRAGCONF` signals, or both sorts of signals arriving out of order. (Bug #33206293)
- A query executed as a pushed join by the `NDB` storage engine returned fewer rows than expected, under the following conditions:
  - The query contained an `IN` or `EXISTS` subquery executed as a pushed join, using the `firstMatch` algorithm.
  - The subquery itself also contained an outer join using at least 2 tables, at least one of which used the `eq_ref` access type.

  (Bug #33181964)

- Part of the work done in NDB 8.0.23 to add query threads to the `ThreadConfig` parameter included the addition of a `TUX` scan context, used to optimize scans, but in some cases this was not set up correctly following the close of a scan. (Bug #33161080, Bug #32794719)

  References: See also: Bug #33379702.

- An `attribute not found` error was returned on a pushed join in `NDB` when looking up a column to add a linked value.

  The issue was caused by use of the wrong lettercase for the name of the column, and is fixed by insuring that we use the unmodified, original name of the column when performing lookups. (Bug #33104337)

- Changes in NDB 8.0 resulted in a permanent error (NDB Error 261) being returned when the resources needed by a transaction's operations did not fit within those allocated for the transaction coordinator, rather than a temporary one (Error 233) as in previous versions. This is significant in NDB Replication, in which a temporary error is retried, but a permanent error is not; a permanent error is suitable when the transaction itself is too large to fit in the transaction coordinator without reconfiguration, but when the transaction cannot fit due to consumption of resources by other transactions, the error should be temporary, as the transaction may be able to fit later, or in some other `TC` instance.

  The temporary error returned in such cases (NDB error 233) now has a slightly different meaning; that is, that there is insufficient pooled memory for allocating another operation. (Previously, this error meant that the limit set by `MaxNoOfConcurrentOperations` had been reached.)

  Rather than conflate these meanings (dynamic allocation and configured limit), we add a new temporary error (Error 234) which is returned when the configured limit has been reached. See Temporary Resource error, and Application error, for more information about these errors. (Bug #32997832)

  References: See also: Bug #33092571.

- Added an `ndbrequire()` in `QMGR` to check whether the node ID received from the `CM_REGREF` signal is less than `MAX_NDB_NODES`. (Bug #32983311)

- A check was reported missing from the code for handling `GET_TABLEID_REQ` signals. To fix this issue, all code relating to all `GET_TABLEID_*` signals has been removed from the `NDB` sources, since these signals are no longer used or supported in NDB Cluster. (Bug #32983249)

- Added an `ndbrequire()` in `QMGR` to ensure that process reports from signal data use appropriate node IDs. (Bug #32983240)

- It was possible in some cases to specify an invalid node type when working with the internal management API. Now the API specifically disallows invalid node types, and defines an “unknown” node type (`NDB_MGM_NODE_TYPE_UNKNOWN`) to cover such cases. (Bug #32957364)

- `NdbReceiver` did not always initialize storage for a MySQL `BIT` column correctly. (Bug #32920099)

- Receiving a spurious schema operation reply from a node not registered as a participant in the current schema operation led to an unplanned shutdown of the SQL node.

  Now in such cases we discard replies from any node not registered as a participant. (Bug #32891206)

  References: See also: Bug #30930132, Bug #32509544.

- The values `true` and `false` for Boolean parameters such as `AutomaticThreadConfig` were not handled correctly when set in a `.cnf` file. (This issue did not affect handling of such values in `.ini` files.) (Bug #32871875)

- Removed unneeded copying of a temporary variable which caused a compiler truncation warning in `storage/ndb/src/common/util/version.cpp`. (Bug #32763321)

- The maximum index size supported by the `NDB` index statistics implementation is 3056 bytes. Attempting to create an index of a larger size when the table held enough data to trigger a statistics update caused `CREATE INDEX` to be rejected with the error `Got error 911 'Index stat scan requested on index with unsupported key size' from NDBCLUSTER`.

  This error originated in the `TUX` kernel block during a scan which caused the schema transaction to fail. This scan is triggered during index creation when the table contains a nonzero number of rows; this also occurs during automatic updates of index statistics or execution of `ANALYZE TABLE`.

  Creating the index as part of `CREATE TABLE` or when the table contained no rows returned no error. No statistics were generated in such situations, while `ANALYZE TABLE` returned an error similar to the one above.

  We fix this by allowing the index to be created while returning an appropriate warning from a new check introduced at the handler level. In addition, the `TUX` scan now handles this situation by suppressing the error, and instead returns success, effectively treating the table as an empty fragment. Otherwise, the behavior in such cases remains unchanged, with a warning returned to the client and no index statistics generated, whether or not the table contains any rows. (Bug #32749829)

  References: This issue is a regression of: Bug #28714864.

- A `CREATE TABLE` statement using ordered indexes returned an error when `IndexStatAutoCreate` was set to `1` and all SQL nodes had been started with `--ndb-index-stat-enable=OFF`, due to the fact that, when set to `OFF`, the option prevented the creation of the index statistics tables. Now these tables are always created at `mysqld` startup regardless of the value of `--ndb-index-stat-enable`. (Bug #32649528)

- If an `NDB` schema operation was lost before the coordinator could process it, the client which logged the operation waited indefinitely for the coordinator to complete or abort it. (Bug #32593352)

  References: See also: Bug #32579148.

- `ndb_mgmd` now writes a descriptive error message to the cluster log when it is invoked with one or more invalid options. (Bug #32554492)

- An IPv6 address used as part of an `NDB` connection string and which had only decimal digits following the first colon was incorrectly parsed, and could not be used to connect to the management server. (Bug #32532157)

- Simultaneously creating a user and then granting this user the `NDB_STORED_USER` privilege on different MySQL servers sometimes caused these servers to hang.

  This was due to the fact that, when the `NDB` storage engine is enabled, all SQL statements that involve users and grants are evaluated to determine whether they effect any users having the `NDB_STORED_USER` privilege, after which some statements are ignored, some are distributed to all SQL nodes as statements, and some are distributed to all SQL nodes as requests to read and apply a user privilege snapshot. These snapshots are stored in the `mysql.ndb_sql_metadata` table. Unlike a statement update, which is limited to one SQL statement, a snapshot update can contain up to seven SQL statements per user. Waiting for any lock in the `NDB` binary logging thread while managing distributed users could easily lead to a deadlock, when the thread was waiting for an exclusive lock on the local ACL cache.

  We fix this problem by implementing explicit locking around `NDB_STORED_USER` snapshot updates; snapshot distribution is now performed while holding a global read lock on one row of the `ndb_sql_metadata` table. (Previously, both statement and snapshot distribution were performed asynchronously, with no locking.) Now, when a thread does not obtain this lock on the first attempt, a warning is raised, and the deadlock prevented.

For more information, see Privilege Synchronization and NDB_STORED_USER. (Bug #32424653)

References: See also: Bug #32832676.

- It was not possible to create or update index statistics when the cluster was in single user mode, due to transactions being disallowed from any node other than the designated API node granted access, regardless of type. This prevented the data node responsible for starting transactions relating to index statistics from doing so.

  We address this issue by relaxing the constraint in single user mode and allowing transactions originating from data nodes (but not from other API nodes). (Bug #32407897)

- When starting multiple management nodes, the first such node waits for the others to start before committing the configuration, but this was not explicitly communicated to users. In addition, when data nodes were started without starting all management nodes, no indication was given to users that its node ID was not allocated since no configuration had yet been committed. Now in such cases, the management node prints a message advising the user that the cluster is configured to use multiple management nodes, and to ensure that all such nodes have been started. (Bug #32339789)

- To handle cases in which a cluster is restarted while the MySQL Server (SQL node) is left running, the index statistics thread is notified when an initial cluster start or restart occurs. The index statistics thread forced the creation of a fresh `Ndb` object and checking of various system objects, which is unnecessary when the MySQL Server is started at the same time as the initial Cluster start which led to the unnecessary re-creation of the `Ndb` object.

  We fix this by restarting only the listener in such cases, rather than forcing the `Ndb` object to be re-created. (Bug #29610555, Bug #33130864)

- Removed extraneous spaces that appeared in some entries written by errors in the node logs. (Bug #29540486)

- `ndb_restore` raised a warning to use `--disable-indexes` when restoring data after the metadata had already been restored with `--disable-indexes`.

  When `--disable-indexes` is used to restore metadata before restoring data, the tables in the target schema have no indexes. We now check when restoring data with this option to ensure that there are no indexes on the target table, and print the warning only if the table already has indexes. (Bug #28749799)

- When restoring of metadata was done using `--disable-indexes`, there was no attempt to create indexes or foreign keys dependent on these indexes, but when `ndb_restore` was used without the option, indexes and foreign keys were created. When `--disable-indexes` was used later while restoring data, `NDB` attempted to drop any indexes created in the previous step, but ignored the failure of a drop index operation due to a dependency on the index of a foreign key which had not been dropped. This led subsequently to problems while rebuilding indexes, when there was an attempt to create foreign keys which already existed.

  We fix `ndb_restore` as follows:

  - When `--disable-indexes` is used, `ndb_restore` now drops any foreign keys restored from the backup.

  - `ndb_restore` now checks for the existence of indexes before attempting to drop them.

  (Bug #26974491)

- The `--ndb-nodegroup-map` option for `ndb_restore` did not function as intended, and code supporting it has been removed. The option now does nothing, and any value set for it is ignored. (Bug #25449055)

- Event buffer status messages shown by the event logger have been improved. Percentages are now displayed only when it makes to do so. In addition, if a maximum size is not defined, the printout shows `max=unlimited`. (Bug #21276857)
- File handles and `FileLogHandler` objects created in `MgmtSrvr::configure_eventlogger` were leaked due to an incomplete destructor for `BufferedLogHandler`. This meant that, each time the cluster configuration changed in a running `ndb_mgmd`, the cluster log was reopened and a file handle leaked, which could lead to issues with test programs and possibly to other problems. (Bug #18192573)
- When `--configdir` was specified as `.`, but with a current working directory other than `DataDir`, the binary configuration was created in `DataDir` and not in the current directory. In addition, `ndb_mgmd` would not start when there was an existing binary configuration in `DataDir`.

  We fix this by having `ndb_mgmd` check the path and refusing to start when a relative path is specified for `--configdir`. (Bug #11755867)
- A memory leak occurred when `NDBCLUSTER` was unable to create a subscription for receiving cluster events. Ownership of the provided event data is supposed to be taken over but actually happened only when creation succeeded, in other cases the provided event data simply being lost. (Bug #102794, Bug #32579459)
- `ndb_mgmd` ignores the `--ndb-connectstring` option if `--config-file` is also specified. Now a warning to this effect is issued, if both options are used. (Bug #102738, Bug #32554759)
- The data node configuration parameters `UndoDataBuffer` and `UndoIndexBuffer` have no effect in any currently supported version of NDB Cluster. Both parameters are now deprecated and the presence of either in the cluster configuration file raises a warning; you should expect them to be removed in a future release. (Bug #84184, Bug #26448357)
- Execution of a bulk `UPDATE` statement using a `LIMIT` clause led to a debug assertion when an error was returned by `NDB`. We fix this by relaxing the assertion for `NDB` tables, since we expect in certain scenarios for an error to be returned at this juncture.

Changes in MySQL NDB Cluster 8.0.26 (2021-07-20, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **NDB Cluster APIs:** The version of `Node.js` used by `NDB` has been upgraded to 12.22.1. (Bug #32640847)
- **NDB Cluster APIs:** Added the `NdbScanFilter::setSqlCmpSemantics()` method to the NDB API. Previously, `NdbScanFilter` has always treated `NULL` as equal to itself, so that `NULL == NULL` evaluates as Boolean `TRUE`; this is not in accordance with the SQL standard, which requires that `NULL == NULL` returns `NULL`. The new method makes it possible to override the traditional behavior, and enforce SQL-compliant `NULL` comparisons instead, for the lifetime of a given `NdbScanFilter` instance.

  For more information, see NdbScanFilter::setSqlCmpSemantics(), in the *MySQL NDB Cluster API Developer Guide*. (WL #14476)
- `ndb_restore` now supports conversion between `NULL` and `NOT NULL` columns, as follows:
  - To restore a `NULL` column as `NOT NULL`, use the `--lossy-conversions` option. The presence of any `NULL` rows in the column causes `ndb_restore` to raise an and exit.
  - To restore a `NOT NULL` column as `NULL`, use the `--promote-attributes` option.

For more information, see the descriptions of the indicated `ndb_restore` options. (Bug #32702637)

- Added the `PreferIPVersion` configuration parameter, which controls the addressing preference of the DNS resolver for IPv4 (4) or IPv6 (6), with 4 being the default. This parameter must be the same for all TCP connections; for this reason, you should set it only in the `[tcp default]` section of the cluster global configuration file. (Bug #32420615)

Bugs Fixed

- **Packaging:** The `ndb-common` man page was removed, and the information it contained moved to other man pages. (Bug #32799519)
- **Packaging:** The `mysqlbinlog` utility was not included in the NDB Cluster Docker image. (Bug #32795044)
- **NDB Cluster APIs:** The Node.js adapter did not always handle character set and collation numbers correctly. (Bug #32742521)
- **NDB Cluster APIs:** Added the `NDB_LE_EventBufferStatus3` log event type to `Ndb_logevent_type` in the MGM API. This is an extension of the `NDB_LE_EventBufferStatus` type which handles total, maximum, and allocated bytes as 64-bit values.

  As part of this fix, the maximum value of the `ndb_eventbuffer_max_alloc` server system variable is increased to 9223372036854775807 (2^{63} - 1).

  For more information, see The Ndb_logevent_type Type. (Bug #32381666)
- Conditions which were pushable to the `NDBCLUSTER` engine were not pushed down to the table if it was referred to as part of a view or a table subquery. (Bug #32924533)
- RPM builds of `NDB` for Docker which used dynamic linking did not complete due to the inclusion of the `ndbclient` library by `ndbxfrm`. Now `ndbxfrm` uses the internal `ndbgeneral` and `ndbportlib` libraries instead.

  As part of this fix, `ndb_restore` also now links against `ndbgeneral` and `ndbportlib`. (Bug #32886430)
- `NDB` now uses `std::min()` and `std::max()` in place of its own internal macros for determining the minimum and maximum of two numbers. (Bug #32854692)
- Some error messages printed by `ndb_restore` tried to access transactions that were already closed for error information, resulting in an unplanned exit. (Bug #32815725)
- The error messages for `NDB` errors 418 (`Out of transaction buffers in LQH...`), 419 (`Out of signal memory...`), and 805 (`Out of attrinfo records in tuple manager...`) all referred to increasing `LongSignalMemory`, although there is no configuration parameter by that name. All three of these error messages have been corrected to refer to the `LongMessageBuffer` parameter instead. (Bug #32797240)
- An unsuccessful `CREATE TABLE` of an `NDB` table returns a generic error message (`ERROR HY000: Can't create table 'tbl'`), with additional, more specific error messages often pushed as warnings. To assist users who may not be aware of this and see only the generic message, we have added reminder text regarding the *`SHOW WARNINGS`* statement to the generic error message, to prompt the user to obtain additional information that might help resolve a given issue more quickly. (Bug #32788231)
- An `NDB` error which is not mapped to a MySQL handler error code is typically presented to a MySQL user as error 1296 or 1297, with a message indicating the underlying `NDB` error code; one exception to this behavior is a `COMMIT` error (originating from `ndbcluster_commit()`), for which the usual `NDB` error is 4350 `Transaction already aborted`. MySQL eventually passed this to

strerror() in the C library, where it was prefixed with Unknown error or similar, but the precise format of this prefix varied with platform-specific differences with the version of libc being used.

We fix this by creating both a new handler error HA_ERR_TX_FAILED, and a new client error ER_TRANSACTION_FAILED, associated with SQL State 25000 Invalid Transaction State. (Bug #32763179)

References: See also: Bug #30264401.

- When started with the --print-full-config option, ndb_mgmd exited with the error Address already in use. This is fixed by skipping free port validation when this option is specified. (Bug #32759903)
- Removed unneeded printouts that were generated in the cluster log when executing queries against the ndbinfo.cluster_locks table. (Bug #32747486)
- The DbUtil class did not call mysql_library_end() when a thread using the MySQL client library had finished doing so, and did not release the thread's local resources by calling mysql_thread_end(). (Bug #32730214)
- A memory leak took place in DbUtil when running a query for the second time using same DbUtil instance; the connection check did not detect the existing MYSQL instance, and replaced it without releasing it. (Bug #32730047)
- Returning an error while determining the number of partitions used by a NDB table caused the MySQL server to write Incorrect information in *table*.frm file to its error log, despite the fact that the indicated file did not exist. This also led to problems with flooding of the error log when users attempted to open NDB tables while the MySQL server was not actually connected to NDB.

  We fix this by changing the function that determines the number of partitions to use the value loaded from the MySQL data dictionary without fetching it from NDB, which also saves one round trip when opening a table. For the special case in which the table is opened for upgrade, we fall back to fetching the value from NDB in the upgrade code path. (Bug #32713166)
- Using duplicate node IDs with CREATE NODEGROUP (for example, CREATE NODEGROUP 11, 11) could lead to an unplanned shutdown of the cluster. Now when this command includes duplicate node IDs, it raises an error. (Bug #32701583)
- Improved the performance of queries against the ndbinfo.cluster_locks table, which could in some cases run quite slowly. (Bug #32655988)
- Fixed a number of issues found in ndb_print_backup_file relating to argument parsing, error reporting, and opening of encrypted files using classes from ndbxfrm. (Bug #32583227)
- The directory unittest/ndb was generated by the build process even though it is not used. This directory is no longer created when building NDB. (Bug #32553339)
- To ensure that the log records kept for the redo log in main memory are written to redo log file within one second, a time supervisor in DBLQH acquires a lock on the redo log part prior to the write. A fix for a previous issue caused a continueB signal (introduced as part of that fix) to be sent when the redo log file was not yet opened and ready for the write, then to return without releasing the lock. Now such cases we release the acquired lock before waiting for the redo log file to be open and ready for the write. (Bug #32539348)

  References: This issue is a regression of: Bug #31585833.

- Updating the `Ndb` object used for receiving events from `NDB` in the binary log injector thread with the value for `ndb_eventbuffer_max_alloc` was performed both at the start of each epoch and after having handled one event, when it is sufficient to update the value once per epoch.

  We fix this by not updating from the global value during processing of each event, which reduces the amount of work required during each event processing loop. (Bug #32494904)

- Failure to find all blob parts for a blob column while reading from the event stream was not handled properly, which caused the data in the caller's copy-out buffer to be incomplete, with no error returned to the caller.

  When a user of the event API has been notified that data has been received for a table with blob column, it creates a buffer large enough to hold the entire blob and then calls the function to read the blob column from the event stream. Most blob types are stored as several small parts in `NDB`; to read the blob data for a blob column from the event stream, the buffered event data must be traversed to find the blob parts and to copy each part into the provided buffer. Each piece of buffered event data associated with the blob column is examined to see whether it contains the data for the blob part desired. When a blob part is found, it is copied into the buffer at the original offset provided by the caller.

  The function which finds the blob parts can copy out one or more blob parts at a time. This function is normally called several times while putting the blob parts together—first to find the first blob part, then all the parts in the middle (several at a time), and then the remainder in the last part. When the function does not find all requested blob parts in the buffered event data, this results in an inconsistency which may occur due to any of several different cases—all parts may not have been sent, the received parts may have been stored in the wrong place, there is a problem in the logic putting the blob parts together, or possibly some other issue. The inconsistency is detected by comparing how many blob parts have been found with how many were requested to be copied out this time.

  This problem was noticed while investigating problem with an unplanned SQL node shutdown that could occur while executing some `ALTER TABLE` operations, where a debug-compiled `mysqld` asserted after having printed information about missing blob parts; manual code inspection shows that a release-compiled binary would just return the incomplete buffer to the caller. This problem was also noticed in addressing some previous similar issues.

  We fix this problem by returning an error from `NdbEventOperationImpl::readBlobParts()` whenever requested blob parts cannot be found. Since this is a serious inconsistency, we also extend the printout provided when this problem is detected. A sample of the extended printout is shown here:

```
= print_blob_part_bufs ============================
part_start: 0, part_count: 15
table: { id: 13, version: 2, name: 't1' }
column: { attrid: 1, name: 'b' }
blob parts table: { id: 14, version: 2, name: 'NDB$BLOB_13_1' }
available buffers: {
[0]*:  part_number: 1, size: 2000, offset: 2000
[1]*:  part_number: 14, size: 2000, offset: 28000
[2]*:  part_number: 7, size: 2000, offset: 14000
[3]*:  part_number: 5, size: 2000, offset: 10000
[4]*:  part_number: 3, size: 2000, offset: 6000
[5]*:  part_number: 0, size: 2000, offset: 0
[6] :  part_number: 15, size: 2000, offset: 30000
[7]*:  part_number: 13, size: 2000, offset: 26000
[8]*:  part_number: 12, size: 2000, offset: 24000
[9]*:  part_number: 11, size: 2000, offset: 22000
[10]*:  part_number: 10, size: 2000, offset: 20000
[11]*:  part_number: 9, size: 2000, offset: 18000
[12]*:  part_number: 8, size: 2000, offset: 16000
[13]*:  part_number: 6, size: 2000, offset: 12000
[14]*:  part_number: 4, size: 2000, offset: 8000
[15]*:  part_number: 2, size: 2000, offset: 4000
}
```

(Bug #32469090)

References: See also: Bug #32405937, Bug #30509284.

- A node was permitted during a restart to participate in a backup before it had completed recovery, instead of being made to wait until its recovery was finished. (Bug #32381165)
- Removed `NDB_WIN32` from the NDB Cluster sources. This define was once intended to demarcate code to be conditionally compiled only for Windows, but had long since been superseded for this purpose by `_WIN32`. (Bug #32380725)
- Running out of disk space while performing an `NDB` backup could lead to an unplanned shutdown of the cluster. (Bug #32367250)
- The index statistics thread relies on the binary log injector thread to inform it about initial system restarts. The index statistics thread then (asynchronously) recycles its `Ndb` object and creates its system tables. Depending on timing, it was possible for the index statistics thread not to be ready to serve requests for a period of time during which `NDB` tables were writable. This also led to issues during the setup of stored grants when the data node parameter `IndexStatAutoCreate` was set to 1.

  We fix this in two ways:

  - Make the sending of the signal to the index statistics thread part of binary log setup so that it is detected in a timely fashion
  - Forcing binary log setup to wait until index statistics functionality has been set up in such cases

  (Bug #32355045)
- It was possible to start `ndb_mgmd` with `NoOfReplicas` set equal to 1 and with more than 72 data nodes defined in the `config.ini` file. Now the management server checks for this condition, and refuses to start if it is found. (Bug #32258207)
- It was possible to start `ndb_mgmd` with an invalid value set in `config.ini` for the `NodeGroup` parameter; subsequently, data node processes using that value were unable to start. Now in such cases, the management server refuses to start, and provides an appropriate error message. (Bug #32210176)
- A statement such `ALTER TABLE t1 ALGORITHM=INPLACE, RENAME COLUMN B to b` that performed an in-place rename of a column changing only the lettercase of its name was successful, but the change was not reflected in the `NDB` dictionary (as shown, for example, in the output of `ndb_desc`). We fix this issue by ensuring that the `NDB` dictionary always matches the lettercase specified in the SQL statement, and that this matches the name as stored in the MySQL data dictionary. (Bug #31958327)
- Event buffer congestion could lead to unplanned shutdown of SQL nodes which were performing binary logging. We fix this by updating the binary logging handler to use `Ndb::pollEvents2()` (rather than the deprecated `pollEvents()` method) to catch and handle such errors properly, instead. (Bug #31926584)
- The `--resume` option for `ndb_import` did not work correctly unless the `--stats` option was also specified. (Bug #31107058)
- Reverted a previous change in the scope of the flags used by `INSERT IGNORE` and other similar SQL statements to inform the handler that duplicate key errors during an insert or update do not stop an ongoing transaction. Now these flags are cleared after every write row event, as before. (Bug #27538524)

  References: See also: Bug #22603412. This issue is a regression of: Bug #20017428.

- NDBCLUSTER uses bitmaps of type MY_BITMAP for keeping track of which columns are to be used in various contexts. When used in short-lived performance-critical code, these are initialized with a bit buffer whose (fixed) size is defined at compile time. The size of these buffers was calculated in multiple ways, which could lead to copy-paste errors, uncertainty whether the buffer is large enough, and possible allocation of excess space.

  We fix this by implementing an internal Ndb_bitmap_buf class that takes the number of bits the buffer should hold as a template argument, and changing all occurrences of static bitmap buffers to instances of Ndb_bitmap_buf. This also saves several bytes in the condition pushdown code in which the buffers were too large. (Bug #27150799)

- The NDB binlog injector thread now detects errors while handling data change events received from the storage engine. If an error is detected, the thread logs error messages and restarts itself, and as part of the restart an exceptional, incident, or LOST_EVENTS entry is written to the binary log. This special entry indicates to a replication applier that the binary log is incomplete. (Bug #18280115, Bug #27150740)

- A DELETE statement whose WHERE clause referred to a BLOB column was not executed correctly. (Bug #13881465)

- Analysis of data node and management node logs was sometimes hampered by the fact that not all log messages included timestamps. This is fixed by replacing a number of different logging functions (printf, fprintf, ndbout, ndbout_c, << overloading, and so on) with and standardizing on the existing EventLogger mechanism which begins each log message with a timestamp in YYYY-MM-DD HH:MM:SS format.

  For more information about NDB Cluster event logs and the log message format, see Event Reports Generated in NDB Cluster. (WL #14311)

  References: See also: Bug #21441915, Bug #30455830.

Changes in MySQL NDB Cluster 8.0.24 (2021-04-20, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **NDB Cluster APIs:** The version of Node.js used by NDB has been upgraded to 12.20.1. (Bug #32356419)

- **ndbinfo Information Database:** Added the dict_obj_tree table to the ndbinfo information database. This table provides information about NDB database objects similar to what is shown by the dict_obj_info table, but presents it in a hierarchical or tree-like fashion that simplifies seeing relationships between objects such as: tables and indexes; tablespaces and data files; log file groups and undo log files.

  An example of such a view of a table t1, having a primary key on column a and a unique key on column b, is shown here:

```
mysql> SELECT indented_name FROM ndbinfo.dict_obj_tree
    -> WHERE root_name = 'test/def/t1';
+----------------------------+
| indented_name              |
+----------------------------+
| test/def/t1                |
|   -> sys/def/13/b          |
|     -> NDB$INDEX_15_CUSTOM |
|   -> sys/def/13/b$unique   |
|     -> NDB$INDEX_16_UI     |
|   -> sys/def/13/PRIMARY    |
```

```
|     -> NDB$INDEX_14_CUSTOM |
+----------------------------+
7 rows in set (0.15 sec)
```

For additional information and examples, see The ndbinfo dict_obj_tree Table. (Bug #32198754)

- **ndbinfo Information Database:** Added the `backup_id` table to the `ndbinfo` information database. This table contains a single column (`id`) and a single row, in which the column value is the backup ID of the most recent backup of the cluster taken with the `ndb_mgm` client. If no NDB backups can be found, the value is 0.

  Selecting from this table replaces the process of obtaining this information by using the `ndb_select_all` utility to dump the contents of the internal `SYSTAB_0` table, which is error-prone and can require an excessively long time to complete. (Bug #32073640)

- Added the status variable `Ndb_config_generation`, which shows the generation number of the current configuration being used by the cluster. This can be used as an indicator to determine whether the configuration of the cluster has changed. (Bug #32247424)

- NDB Cluster now uses the MySQL `host_application_signal` component service to perform shutdown of SQL nodes. (Bug #30535835, Bug #32004109)

- `NDB` has implemented the following two improvements in calculation of index statistics:

  - Previously, index statistics were collected from a single fragment only; this is changed such that additional fragments are used for these.

  - The algorithm used for very small tables, such as those having very few rows where results are discarded, has been improved, so that estimates for such tables should be more accurate than previously.

  See NDB API Statistics Counters and Variables, for more information. (WL #13144)

- A number of NDB Cluster programs now support input of the password for encrypting or decrypting an `NDB` backup from standard input. Changes relating to each program affected are listed here:

  - For `ndb_restore`, the `--backup-password-from-stdin` option introduced in this release enables input of the password in a secure fashion, similar to how it is done by the `mysql` client' `--password` option. Use this option together with the `--decrypt` option.

  - `ndb_print_backup_file` now also supports `--backup-password-from-stdin` as the long form of the existing `-P` option.

  - For `ndb_mgm`, `--backup-password-from-stdin` is supported together with `--execute "START BACKUP [options]"` for starting an encrypted cluster backup from the system shell, and has the same effect.

  - Two options for `ndbxfrm`, `--encrypt-password-from-stdin` and `--decrypt-password-from-stdin`, which are also introduced in this release, cause similar behavior when using this program, respectively, to encrypt or to decrypt a backup file.

  In addition, you can cause `ndb_mgm` to use encryption whenever it creates a backup by starting it with `--encrypt-backup`. In this case, the user is prompted for a password when invoking `START BACKUP` if none is supplied. This option can also be specified in the `[ndb_mgm]` section of the `my.cnf` file.

  Also, the behavior and syntax of the `ndb_mgm` management client `START BACKUP` are changed slightly, such that it is now possible to use the `ENCRYPT` option without also specifying `PASSWORD`. Now when the user does this, the management client prompts the user for a password.

  For more information, see the descriptions of the NDB Cluster programs and program options just mentioned, as well as Online Backup of NDB Cluster. (WL #14259)

Bugs Fixed

- **Packaging:** The `mysql-cluster-community-server-debug` and `mysql-cluster-commercial-server-debug` RPM packages were dependent on `mysql-community-server` and `mysql-commercial-server`, respectively, `instead of mysql-cluster-community-server` and `mysql-cluster-commercial-server`. (Bug #32683923)

- **Packaging:** RPM upgrades from NDB 7.6.15 to 8.0.22 did not succeed due to a file having been moved from the `server` RPM to the `client-plugins` RPM. (Bug #32208337)

- **Linux:** On Linux systems, `NDB` interpreted memory sizes obtained from `/proc/meminfo` as being supplied in bytes rather than kilobytes. (Bug #102505, Bug #32474829)

- **Microsoft Windows:** Removed several warnings which were generated when building NDB Cluster on Windows using Microsoft Visual Studio 2019. (Bug #32107056)

- **Microsoft Windows:** `NDB` failed to start correctly on Windows when initializing the `NDB` library with `ndb_init()`, with the error `Failed to find CPU in CPU group`.

  This issue was due to how Windows works with regard to assigning processes to CPUs: when there are more than 64 logical CPUs on a machine, Windows divides them into different processor groups during boot. Each processor group can at most hold 64 CPUs; by default, a process can be assigned to only one processor group. The function `std::thread::hardware_concurrency()` was used to get the maximum number of logical CPUs on the machine, but on Windows, this function returns only the maximum number of logical CPUs present in the processor group with which the current process is affiliated. This value is used to allocate memory for an array that holds hardware information about each CPU on the machine. Since the array held valid memory for CPUs from only one processor group, any attempt to store and retrieve hardware information about a CPU in a different processor group led to array bound read/write errors, leading to memory corruption and ultimately leads to process failures.

  Fixed by using `GetActiveProcessorCount()` instead of the `hardware_concurrency()` function referenced previously. (Bug #101347, Bug #32074703)

- **Solaris:** While preparing `NDBFS` for handling of encrypted backups, activation of `O_DIRECT` was suspended until after initialization of files was completed. This caused initialization of redo log files to require an excessive amount of time on systems using hard disk drives with `ext3` file systems.

  On Solaris, `directio` is used instead of `O_DIRECT`; activating `directio` prior to initialization of files caused a notable increase in time required when using hard disk drives with `UFS` file systems.

  Now we ensure that, on systems having `O_DIRECT`, this is activated before initialization of files, and that, on Solaris, `directio` continues to be activated after initialization of files. (Bug #32187942)

- **NDB Cluster APIs:** Several NDB API coding examples included in the source did not release all resources allocated. (Bug #31987735)

- **NDB Cluster APIs:** Some internal dictionary objects in `NDB` used an internal name format which depends on the database name of the `Ndb` object. This dependency has been made more explicit where necessary and otherwise removed.

  Users of the NDB API should be aware that the `fullyQualified` argument to `Dictionary::listObjects()` still works in such a way that specifying it as `false` causes the objects in the list it returns to use fully qualified names. (Bug #31924949)

- **ndbinfo Information Database:** The system variables `ndbinfo_database` and `ndbinfo_table_prefix` are intended to be read-only. It was found that it was possible to set `mysqld` command-line options corresponding to either or both of these; doing so caused the `ndbinfo` database to malfunction. This fix insures that it is no longer possible to set either of these variables in the `mysql` client or from the command line. (Bug #23583256)

- In some cases, a query affecting a user with the `NDB_STORED_USER` privilege could be printed to the MySQL server log without being rewritten. Now such queries are omitted or rewritten to remove any text following the keyword `IDENTIFIED`. (Bug #32541096)
- The value set for the `SpinMethod` data node configuration parameter was ignored. (Bug #32478388)
- The compile-time debug flag `DEBUG_FRAGMENT_LOCK` was enabled by default. This caused increased resource usage by `DBLQH`, even for release builds.

  This is fixed by disabling `DEBUG_FRAGMENT_LOCK` by default. (Bug #32459625)
- `ndb_mgmd` now exits gracefully in the event of a `SIGTERM` just as it does following a management client `SHUTDOWN` command. (Bug #32446105)
- When started on a port which was already in use, `ndb_mgmd` did not throw any errors since the use of `SO_REUSEADDR` on Windows platforms allowed multiple sockets to bind to the same address and port.

  To take care of this issue, we replace `SO_REUSEADDRPORT` with `SO_EXCLUSIVEADDRUSE`, which prevents re-use of a port that is already in use. (Bug #32433002)
- Encountering an error in detection of an initial system restart of the cluster caused the SQL node to exit prematurely. (Bug #32424580)
- Under some situations, when trying to measure the time of a CPU pause, an elapsed time of zero could result. In addition, computing the average for a very fast spin (for example, 100 loops taking less than 100ns) could zero nanoseconds. In both cases, this caused the spin calibration algorithm throw an arithmetic exception due to division by zero.

  We fix both issues by modifying the algorithm so that it ignores zero values when computing mean spin time. (Bug #32413458)

  References: See also: Bug #32497174.
- Table and database names were not formatted correctly in the messages written to the `mysqld` error log when the internal method `Ndb_rep_tab_reader::scan_candidates()` found ambiguous matches for a given database, table, or server ID in the `ndb_replication` table. (Bug #32393245)
- Some queries with nested pushed joins were not processed correctly. (Bug #32354817)
- When `ndb_mgmd` allocates a node ID, it reads through the configuration to find a suitable ID, causing a mutex to be held while performing hostname lookups. Because network address resolution can require large amounts of time, it is not considered good practice to hold such a mutex or lock while performing network operations.

  This issue is fixed by building a list of configured nodes while holding the mutex, then using the list to perform hostname matching and other logic. (Bug #32294679)
- The schema distribution participant failed to start a global checkpoint after writing a reply to the `ndb_schema_result` table, which caused an unnecessary delay before the coordinator received events from the participant notifying it of the result. (Bug #32284873)
- The global DNS cache used in `ndb_mgmd` caused stale lookups when restarting a node on a new machine with a new IP address, which meant that the node could not allocate a node ID.

  This issue is addressed by the following changes:

  - Node ID allocation no longer depends on `LocalDnsCache`
  - `DnsCache` now uses local scope only

  (Bug #32264914)

- `ndb_restore` generated a core file when started with unknown or invalid arguments. (Bug #32257374)
- Auto-synchronization detected the presence of mock foreign key tables in the NDB dictionary and attempted to re-create them in the MySQL server's data dictionary, although these should remain internal to the NDB Dictionary and not be exposed to the MySQL server. To fix this issue, we now ensure that the NDB Cluster auto-synchronization mechanism ignores any such mock tables. (Bug #32245636)
- Improved resource usage associated with handling of cluster configuration data. (Bug #32224672)
- Removed left-over debugging printouts from `ndb_mgmd` showing a client's version number upon connection. (Bug #32210216)

  References: This issue is a regression of: Bug #30599413.
- The backup abort protocol for handling of node failures did not function correctly for single-threaded data nodes (`ndbd`). (Bug #32207193)
- While retrieving sorted results from a pushed-down join using `ORDER BY` with the `index` access method (and without `filesort`), an SQL node sometimes unexpectedly terminated. (Bug #32203548)
- Logging of redo log initialization showed log part indexes rather than log part numbers. (Bug #32200635)
- Signal data was overwritten (and lost) due to use of extended signal memory as temporary storage. Now in such cases, extended signal memory is not used in this fashion. (Bug #32195561)
- When `ClassicFragmentation = 1`, the default number of partitions per node (shown in `ndb_desc` output as `PartitionCount`) is calculated using the lowest number of LDM threads employed by any single live node, and was done only once, even after data nodes left or joined the cluster, possibly with a new configuration changing the LDM thread count and thus the default partition count. Now in such cases, we make sure the default number of partitions per node is recalculated each time data nodes join or leave the cluster.

  This is not an issue in NDB 8.0.23 and later, when `ClassicFragmentation` is set to 0. (Bug #32183985)
- The internal function `Ndb_ReloadHWInfo()` is responsible for updating hardware information for all the CPUs on the host. For the Linux ARM platform, which does not have Level 3 cache information, this assigned a socket ID for the L3 cache ID but failed to record the value for the global variable `num_shared_l3_caches`, which is needed when creating lists of CPUs connected to a shared L3 cache. (Bug #32180383)
- When trying to run two management nodes on the same host and using the same port number, it was not always obvious to users why they did not start. Now in such cases, in addition to writing a message to the error log, an error message `Same port number is specified for management nodes node_id1 and node_id2 (or) they both are using the default port number on same host host_name` is also written to the console, making the source of the issue more immediately apparent. (Bug #32175157)
- Added a `--cluster-config-suffix` option for `ndb_mgmd` and `ndb_config`, for use in internal testing to override a defaults group suffix. (Bug #32157276)
- The management server returned the wrong status for host name matching when some of the host names in configuration did not resolve and client trying to allocate a node ID connected from the host whose host name resolved to a loopback address with the error `Could not alloc node id at <host>:<port>: Connection with id X done from wrong host ip 127.0.0.1, expected <unresolvable_host> (lookup failed)`.

  This caused the connecting client to fail the node ID allocation.

This issue is fixed by rewriting the internal match_hostname() function so that it contains all logic for how the requesting client address should match the configured hostnames, and so that it first checks whether the configured host name can be resolved; if not, it now returns a special error so that the client receives an error indicating that node ID allocation can be retried. The new error is `Could not alloc node id at <host>:<port>: No configured host found of node type <type> for connection from ip 127.0.0.1. Some hostnames are currently unresolvable. Can be retried`. (Bug #32136993)

- The internal function `ndb_socket_create_dual_stack()` did not close a newly created socket when a call to `ndb_setsockopt()` was unsuccessful. (Bug #32105957)

- The local checkpoint (LCP) mechanism was changed in NDB 7.6 such that it also detected idle fragments—that is, fragments which had not changed since the last LCP and thus required no on-disk metadata update. The LCP mechanism could then immediately proceed to handle the next fragment. When there were a great many such idle fragments, the CPU consumption required merely to loop through these became highly significant, causing latency spikes in user transactions.

  A 1 ms delay was already inserted between each such idle fragment being handled. Testing later showed this to be too short an interval, and that we are normally not in as great a hurry to complete these idle fragments as we previously believed.

  This fix extends the idle fragment delay time to 20 ms if there are no redo alerts indicating an urgent need to complete the LCP. In case of a low redo alert state we wait 5 ms instead, and for a higher alert state we fall back to the 1 ms delay. (Bug #32068551)

  References: See also: Bug #31655158, Bug #31613158.

- When an `NDB` table was created, it was invalidated in the global dictionary cache, but this was unnecessary. Furthermore, having a table which exists in the global dictionary cache is actually an advantage for subsequent uses of the new table, since it can be found in the table cache without performing a round trip to `NDB`. (Bug #32047456)

- No clear error message was provided when an `ndb_mgmd` process tried to start using the `PortNumber` of a port that was already in use. (Bug #32045786)

- Two problems occurred when `NDB` closed a table:

  - `NDB` failed to detect when the close was done from `FLUSH TABLES`, which meant that the NDB table definitions in the global dictionary cache were not invalidated.

  - When the close was done by a thread which had not used `NDB` earlier—for example when `FLUSH TABLES` or `RESET MASTER` closed instances of `ha_ndbcluster` held in the table definition cache—a new `Thd_ndb` object was allocated, even though there is a fallback to the global `Ndb` object in case the allocation fails, which never occurs in such cases, so it is less wasteful simply to use the global object already provided.

  (Bug #32018394, Bug #32357856)

- Removed a large number of compiler warnings relating to unused function arguments in `NdbDictionaryImpl`. (Bug #31960757)

- Unnecessary casts were performed when checking internal error codes. (Bug #31930166)

- `NDB` continued to use file system paths for determining the names of tables to open or perform DDL on, in spite of the fact that it longer actually uses files for these operations. This required unnecessary translation between character sets, handling the MySQL-specific file system encoding, and parsing. In addition, results of these operations were stored in buffers of fixed size, each instance of which used several hundred bytes of memory unnecessarily. Since the database and table names to use are already available to `NDB` through other means, this translation could be (and has been) removed in most cases. (Bug #31846478)

- Generation of internal statistics relating to NDB object counts was found to lead to an increase in transaction latency at very high rates of transactions per second, brought about by returning an excessive number of freed NDB objects. (Bug #31790329)

- NDB behaved unpredictably in response an attempt to change permissions on a distributed user (that is, a user having the NDB_STORED_USER privilege) during a binary log thread shutdown and restart. We address this issue by ensuring that the user gets a clear warning Could not distribute ACL change to other MySQL servers whenever distribution does not succeed. This fix also improves a number of mysqld log messages. (Bug #31680765)

- ndb_restore encountered intermittent errors while replaying backup logs which deleted blob values; this was due to deletion of blob parts when a main table row containing blob one or more values was deleted. This is fixed by modifying ndb_restore to use the asynchronous API for blob deletes, which does not trigger blob part deletes when a blob main table row is deleted (unlike the synchronous API), so that a delete log event for the main table deletes only the row from the main table. (Bug #31546136)

- When a table creation schema transaction is prepared, the table is in TS_CREATING state, and is changed to TS_ACTIVE state when the schema transaction commits on the DBDIH block. In the case where the node acting as DBDIH coordinator fails while the schema transaction is committing, another node starts taking over for the coordinator. The following actions are taken when handling this node failure:

  - DBDICT rolls the table creation schema transaction forward and commits, resulting in the table involved changing to TS_ACTIVE state.

  - DBDIH starts removing the failed node from tables by moving active table replicas on the failed node from a list of stored fragment replicas to another list.

  These actions are performed asynchronously many times, and when interleaving may cause a race condition. As a result, the replica list in which the replica of a failed node resides becomes nondeterministic and may differ between the recovering node (that is, the new coordinator) and other DIH participant nodes. This difference violated a requirement for knowing which list the failed node's replicas can be found during the recovery of the failed node recovery on the other participants.

  To fix this, moving active table replicas now covers not only tables in TS_ACTIVE state, but those in TS_CREATING (prepared) state as well, since the prepared schema transaction is always rolled forward.

  In addition, the state of a table creation schema transaction which is being aborted is now changed from TS_CREATING or TS_IDLE to TS_DROPPING, to avoid any race condition there. (Bug #30521812)

- START BACKUP SNAPSHOTSTART WAIT STARTED could return control to the user prior to the backup's restore point from the user point of view; that is the Backup started notification was sent before waiting for the synchronising global checkpoint (GCP) boundary. This meant that transactions committed after receiving the notification might be included in the restored data.

  To fix this problem, START BACKUP now sends a notification to the client that the backup has been started only after the GCP has truly started. (Bug #29344262)

- Upgrading to NDB Cluster 8.0 from a prior release includes an upgrade in the schema distribution mechanism, as part of which the ndb_schema table is dropped and recreated in a way which causes all MySQL Servers connected to the cluster to restart their binary log injector threads, causing a gap event to be written to the binary log. Since the thread restart happens at the same time on all MySQL Servers, no binary log spans the time during which the schema distribution functionality upgrade was performed, which breaks NDB Cluster Replication.

  This issue is fixed by adding support for gracefully reconstituting the schema distribution tables while allowing the injector thread to continue processing changes from the cluster. This is implemented

by handling the DDL event notification for `DROP TABLE` to turn off support for schema distribution temporarily, and to start regular checks to re-create the tables. When the tables have been successfully created again, the regular checks are turned off and support for schema distribution is turned back on.

`NDB` also now detects automatically when the `ndb_apply_status` table has been dropped and re-creates it. The drop and re-creation leaves a gap event in the binary log, which in a replication setup causes the replica MySQL Server to stop applying changes from the source until the replication channel is restarted (see ndb_apply_status Table).

In addition, the minimum version required to perform the schema distribution upgrade is raised to 8.0.24, which prevents automatic triggering of the schema distribution upgrade until all connected API nodes support the new upgrade procedure.

For more information, see NDB Cluster Replication Schema and Tables. (Bug #27697409, Bug #30877233)

References: See also: Bug #30876990.

- Fixed a number of issues uncovered when trying to build `NDB` with GCC 6. (Bug #25038373)
- Calculation of the redo alert state based on redo log usage was overly aggressive, and thus incorrect, when using more than 1 log part per LDM.

Changes in MySQL NDB Cluster 8.0.23 (2021-01-18, General Availability)

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

Deprecation and Removal Notes

- **Important Change:** As part of the terminology changes begun in MySQL 8.0.21 and NDB 8.0.21, the `ndb_slave_conflict_role` system variable is now deprecated, and is being replaced with `ndb_conflict_role`.

  In addition, a number of status variables have been deprecated and are being replaced, as shown in the following table:

  Table 2 Deprecated NDB status variables and their replacements

Deprecated variable	Replacement
`Ndb_api_adaptive_send_deferred_count_slave`	`Ndb_api_adaptive_send_deferred_count_replica`
`Ndb_api_adaptive_send_forced_count_slave`	`Ndb_api_adaptive_send_forced_count_replica`
`Ndb_api_adaptive_send_unforced_count_slave`	`Ndb_api_adaptive_send_unforced_count_replica`
`Ndb_api_bytes_received_count_slave`	`Ndb_api_bytes_received_count_replica`
`Ndb_api_bytes_sent_count_slave`	`Ndb_api_bytes_sent_count_replica`
`Ndb_api_pk_op_count_slave`	`Ndb_api_pk_op_count_replica`
`Ndb_api_pruned_scan_count_slave`	`Ndb_api_pruned_scan_count_replica`
`Ndb_api_range_scan_count_slave`	`Ndb_api_range_scan_count_replica`
`Ndb_api_read_row_count_slave`	`Ndb_api_read_row_count_replica`
`Ndb_api_scan_batch_count_slave`	`Ndb_api_scan_batch_count_replica`
`Ndb_api_table_scan_count_slave`	`Ndb_api_table_scan_count_replica`
`Ndb_api_trans_abort_count_slave`	`Ndb_api_trans_abort_count_replica`

Deprecated variable	Replacement
`Ndb_api_trans_close_count_slave`	`Ndb_api_trans_close_count_replica`
`Ndb_api_trans_commit_count_slave`	`Ndb_api_trans_commit_count_replica`
`Ndb_api_trans_local_read_row_count_slave`	`Ndb_api_trans_local_read_row_count_replica`
`Ndb_api_trans_start_count_slave`	`Ndb_api_trans_start_count_replica`
`Ndb_api_uk_op_count_slave`	`Ndb_api_uk_op_count_replica`
`Ndb_api_wait_exec_complete_count_slave`	`Ndb_api_wait_exec_complete_count_replica`
`Ndb_api_wait_meta_request_count_slave`	`Ndb_api_wait_meta_request_count_replica`
`Ndb_api_wait_nanos_count_slave`	`Ndb_api_wait_nanos_count_replica`
`Ndb_api_wait_scan_result_count_slave`	`Ndb_api_wait_scan_result_count_replica`
`Ndb_slave_max_replicated_epoch`	`Ndb_replica_max_replicated_epoch`

Also as part of this work, the `ndbinfo.table_distribution_status` table's `tab_copy_status` column values `ADD_TABLE_MASTER` and `ADD_TABLE_SLAVE` are deprecated, and replaced by, respectively, `ADD_TABLE_COORDINATOR` and `ADD_TABLE_PARTICIPANT`.

Finally, the `--help` output of some NDB utility programs such as `ndb_restore` has been updated. (Bug #31571031)

- **NDB Client Programs:** Effective with this release, the MySQL NDB Cluster Auto-Installer (`ndb_setup.py`) has been has been removed from the NDB Cluster binary and source distributions, and is no longer supported. (Bug #32084831)

  References: See also: Bug #31888835.

- **ndbmemcache:** `ndbmemcache`, which was deprecated in the previous release of NDB Cluster, has now been removed from NDB Cluster, and is no longer supported. (Bug #32106576)

Functionality Added or Changed

- As part of work previously done in NDB 8.0, the metadata check performed as part of auto-synchronization between the representation of an `NDB` table in the NDB dictionary and its counterpart in the MySQL data dictionary has been extended to include, in addition to table-level properties, the properties of columns, indexes, and foreign keys. (This check is also made by a debug MySQL server when performing a `CREATE TABLE` statement, and when opening an `NDB` table.)

  As part of this work, any mismatches found between an object's properties in the NDB dictionary and the MySQL data dictionary are now written to the MySQL error log. The error log message includes the name of the property, its value in the NDB dictionary, and its value in the MySQL data dictionary. If the object is a column, index, or foreign key, the object's type is also indicated in the message. (WL #13412)

- The `ThreadConfig` parameter has been extended with two new thread types, query threads and recovery threads, intended for scaleout of LDM threads. The number of query threads must be a multiple of the number of LDM threads, up to a maximum of 3 times the number of LDM threads.

  It is also now possible when setting `ThreadConfig` to combine the `main` and `rep` threads into a single thread by setting either or both of these arguments to 0.

  When one of these arguments is set to 0 but the other remains set to 1, the resulting combined thread is named `main_rep`. When both are set to 0, they are combined with the `recv` thread (assuming that `recv` to 1), and this combined thread is named `main_rep_recv`. These thread names are those shown when checking the `threads` table in the `ndbinfo` information database.

  In addition, the maximums for a number of existing thread types have been increased. The new maximums are: LDM threads: 332; TC threads: 128; receive threads: 64; send threads: 64; main

threads: 2. (The maximums for query threads and recovery threads are 332 each.) Maximums for other thread types remain unchanged from previous NDB Cluster releases.

Another change related to this work causes NDB to employ mutexes for protecting job buffers when more than 32 block threads are in use. This may cause a slight decrease in performance (roughly 1 to 2 percent), but also results in a decrease in the amount of memory used by very large configurations. For example, a setup with 64 threads which used 2 GB of job buffer memory previously should now require only about 1 GB instead. In our testing this has resulted in an overall improvement (on the order of 5 percent) in the execution of very complex queries.

For more information, see the descriptions of the arguments to the ThreadConfig parameter discussed previously, and of the ndbinfo.threads table. (WL #12532, WL #13219, WL #13338)

- This release adds the possibility of configuring the threads for multithreaded data nodes (ndbmtd) automatically by implementing a new data node configuration parameter AutomaticThreadConfig. When set to 1, NDB sets up the thread assignments automatically, based on the number of processors available to applications. If the system does not limit the number of processors, you can do this by setting NumCPUs to the desired number. Automatic thread configuration makes it unnecessary to set any values for ThreadConfig or MaxNoOfExecutionThreads in config.ini; if AutomaticThreadConfig is enabled, settings for either of these parameters are not used.

  As part of this work, a set of tables providing information about hardware and CPU availability and usage by NDB data nodes have been added to the ndbinfo information database. These tables, along with a brief description of the information provided by each, are listed here:

  - cpudata: CPU usage during the past second
  - cpudata_1sec: CPU usage per second over the past 20 seconds
  - cpudata_20sec: CPU usage per 20-second interval over the past 400 seconds
  - cpudata_50ms: CPU usage per 50-millisecond interval during the past second
  - cpuinfo: The CPU on which the data node executes
  - hwinfo: The hardware on the host where the data node resides

  Not all of the tables listed are available on all platforms supported by NDB Cluster:

  - The cpudata, cpudata_1sec, cpudata_20sec, and cpudata_50ms tables are available only on Linux and Solaris operating systems.
  - The cpuinfo table is not available on FreeBSD or macOS.

  (WL #13980)

- Added statistical information in the DBLQH block which is employed to track the use of key lookups and scans, as well as tracking queries from DBTC and DBSPJ. By detecting situations in which the load is high, but in which there is not actually any need to decrease the number of rows scanned per realtime break, rather than checking the size of job buffer queues to decide how many rows to scan, this makes it possible to scan more rows when there is no CPU congestion. This helps improve performance and realtime behaviour when handling high loads. (WL #14081)

- A new method for handling table partitions and fragments is introduced, such that the number of local data managers (LDMs) for a given data node can determined independently of the number of redo log parts, and that the number of LDMs can now be highly variable. NDB employs this method when the ClassicFragmentation data node configuration parameter, implemented as part of this work, is set to false. When this is done, the number of LDMs is no longer used to determine how many partitions to create for a table per data node; instead, the PartitionsPerNode parameter, also

introduced in this release, now determines this number, which is now used for calculating how many fragments a table should have.

When `ClassicFragmentation` has its default value `true`, then the traditional method of using the number of LDMs is used to determine how many fragments a table should have.

For more information, see Multi-Threading Configuration Parameters (ndbmtd). (WL #13930, WL #14107)

Bugs Fixed

- **macOS:** Removed a number of compiler warnings which occurred when building `NDB` for Mac OS X. (Bug #31726693)
- **Microsoft Windows:** Removed a compiler warning `C4146: unary minus operator applied to unsigned type, result still unsigned` from Visual Studio 2013 found in `storage\ndb\src\kernel\blocks\dbacc\dbaccmain.cpp`. (Bug #23130016)
- **Solaris:** Due to a source-level error, `atomic_swap_32()` was supposed to be specified but was not actually used for Solaris builds of NDB Cluster. (Bug #31765608)
- **NDB Cluster APIs:** Removed redundant usage of `strlen()` in the implementation of `NdbDictionary` and related internal classes in the NDB API. (Bug #100936, Bug #31930362)
- **MySQL NDB ClusterJ:** When a `DomainTypeHandler` was instantiated by a `SessionFactory`, it was stored locally in a static map, `typeToHandlerMap`. If multiple, distinct `SessionFactories` for separate connections to the data nodes were obtained by a ClusterJ application, the static `typeToHandlerMap` would be shared by all those factories. When one of the `SessionFactories` was closed, the connections it created were closed and any tables opened by the connections were cleared from the NDB API global cache. However, the `typeToHandlerMap` was not cleared, and through it the other `SessionFactories` keep accessing the `DomainTypeHandlers` of tables that had already been cleared. These obsolete `DomainTypeHandlers` contained invalid `NdbTable` references and any `ndbapi` calls using those table references ended up with errors.

  This patch fixes the issue by making the `typeToHandlerMap` and the related `proxyInterfacesToDomainClassMap` maps local to a `SessionFactory`, so that they are cleared when the `SessionFactory` is closed. (Bug #31710047)
- **MySQL NDB ClusterJ:** Setting `com.mysql.clusterj.connection.pool.size=0` made connections to an NDB Cluster fail. With this fix, setting `com.mysql.clusterj.connection.pool.size=0` disables connection pooling as expected, so that every request for a `SessionFactory` results in the creation of a new factory and separate connections to the cluster can be created using the same connection string. (Bug #21370745, Bug #31721416)
- When calling `disk_page_abort_prealloc()`, the callback from this internal function is ignored, and so removal of the operation record for the `LQHKEYREQ` signal proceeds without waiting. This left the table subject to removal before the callback had completed, leading to a failure in `PGMAN` when the page was retrieved from disk.

  To avoid this, we add an extra usage count for the table especially for this page cache miss; this count is decremented as soon as the page cache miss returns. This means that we guarantee that the table is still present when returning from the disk read. (Bug #32146931)
- When a table was created, it was possible for a fragment of the table to be checkpointed too early during the next local checkpoint. This meant that Prepare Phase LCP writes were still being performed when the LCP completed, which could lead to problems with subsequent `ALTER TABLE` statements on the table just created. Now we wait for any potential Prepare Phase LCP writes to finish before the LCP is considered complete. (Bug #32130918)

- Using the maximum size of an index key supported by index statistics (3056 bytes) caused buffer issues in data nodes. (Bug #32094904)

  References: See also: Bug #25038373.

- NDB now prefers CLOCK_MONOTONIC which on Linux is adjusted by frequency changes but is not updated during suspend. On macOS, NDB instead uses CLOCK_UPTIME_RAW which is the same, except that it is not affected by any adjustments.

  In addition, when intializing NdbCondition the monotonic clock to use is taken directly from NdbTick, rather than re-executing the same preprocessor logic used by NdbTick. (Bug #32073826)

- ndb_restore terminated unexpectedly when run with the --decrypt option on big-endian systems. (Bug #32068854)

- When the data node receive thread found that the job buffer was too full to receive, nothing was done to ensure that, the next time it checked, it resumed receiving from the transporter at the same point at which it stopped previously. (Bug #32046097)

- The metadata check failed during auto-synchronization of tables restored using the ndb_restore tool. This was a timing issue relating to indexes, and was found in the following two scenarios encountered when a table had been selected for auto-synchronization:

  1. When the indexes had not yet been created in the NDB dictionary

  2. When the indexes had been created, but were not yet usable

  (Bug #32004637)

- Optimized sending of packed signals by registering the kernel blocks affected and the sending functions which need to be called for each one in a data structure rather than looking up this information each time. (Bug #31936941)

- When two data definition language statements—one on a database and another on a table in the same schema—were run in parallel, it was possible for a deadlock to occur. The DDL statement affecting the database acquired the global schema lock first, but before it could acquire a metadata lock on the database, the statement affecting the table acquired an intention-exclusive metadata lock on the schema. The table DDL statement was thus waiting for the global schema lock to upgrade its metadata lock on the table to an exclusive lock, while the database DDL statement waited for an exclusive metadata lock on the database, leading to a deadlock.

  A similar type of deadlock involving tablespaces and tables was already known to occur; NDB already detected and resolved that issue. The current fix extends that logic to handle databases and tables as well, to resolve the problem. (Bug #31875229)

- Clang 8 raised a warning due to an uninitialized variable. (Bug #31864792)

- An empty page acquired for an insert did not receive a log sequence number. This is necessary in case the page was used previously and thus required undo log execution before being used again. (Bug #31859717)

- No reason was provided when rejecting an attempt to perform an in-place ALTER TABLE ... ADD PARTITION statement on a fully replicated table. (Bug #31809290)

- When the master node had recorded a more recent GCI than a node starting up which had performed an unsuccessful restart, subsequent restarts of the latter could not be performed because it could not restore the stated GCI. (Bug #31804713)

- When using 3 or 4 fragment replicas, it is possible to add more than one node at a time, which means that DBLQH and DBDIH can have distribution keys based on numbers of fragment replicas that differ by up to 3 (that is, MAX_REPLICAS - 1), rather than by only 1. (Bug #31784934)

- It was possible in DBLQH for an ABORT signal to arrive from DBTC before it received an LQHKEYREF signal from the next local query handler. Now in such cases, the out-of-order ABORT signal is ignored. (Bug #31782578)
- NDB did not handle correctly the case when an ALTER TABLE ... COMMENT="..." statement did not specify ALGORITHM=COPY. (Bug #31776392)
- It was possible in some cases to miss the end point of undo logging for a fragment. (Bug #31774459)
- ndb_print_sys_file did not work correctly with version 2 of the sysfile format that was introduced in NDB 8.0.18. (Bug #31726653)

  References: See also: Bug #31828452.
- DBLQH could not handle the case in which identical operation records having the same transaction ID came from different transaction coordinators. This led to locked rows persisting after a node failure, which kept node recovery from completing. (Bug #31726568)
- It is possible for DBDIH to receive a local checkpoint having a given ID to restore while a later LCP is actually used instead, but when performing a partial LCP in such cases, the DIH block was not fully synchronized with the ID of the LCP used. (Bug #31726514)
- In most cases, when searching a hash index, the row is used to read the primary key, but when the row has not yet been committed the primary key may be read from the copy row. If the row has been deleted, it can no longer be used to read the primary key. Previously in such cases, the primary key was treated as a NULL, but this could lead to making a comparison using uninitialised data.

  Now when this occurs, the comparison is made only if the row has not been deleted; otherwise the row is checked of among the operations in the serial queue. If no operation has the primary key, then any comparison can be reported as not equal, since no entry in the parallel queue can reinsert the row. This needs to be checked due to the fact that, if an entry in the serial queue is an insert then the primary key from this operation must be identified as such to preclude inserting the same primary key twice. (Bug #31688797)
- As with writing redo log records, when the file currently used for writing global checkpoint records becomes full, writing switches to the next file. This switch is not supposed to occur until the new file is actually ready to receive the records, but no check was made to ensure that this was the case. This could lead to an unplanned data node shutdown restoring data from a backup using ndb_restore. (Bug #31585833)
- Release of shared global memory when it is no longer required by the DBSPJ block now occurs more quickly than previously. (Bug #31321518)

  References: See also: Bug #31231286.
- Stopping 3 nodes out of 4 in a single node group using kill -9 caused an unplanned cluster shutdown. To keep this from happening under such conditions, NDB now ensures that any node group that has not had any node failures is viewed by arbitration checks as fully viable. (Bug #31245543)
- Multi-threaded index builds could sometimes attempt to use an internal function disallowed to them. (Bug #30587462)
- While adding new data nodes to the cluster, and while the management node was restarting with an updated configuration file, some data nodes terminated unexpectedly with the error virtual void TCP_Transporter::resetBuffers(): Assertion `!isConnected()' failed. (Bug #30088051)
- It was not possible to execute TRUNCATE TABLE or DROP TABLE for the parent table of a foreign key with foreign_key_checks set to 0. (Bug #97501, Bug #30509759)

- Optimized the internal `NdbReceiver::unpackNdbRecord()` method, which is used to convert rows retrieved from the data nodes from packed wire format to the NDB API row format. Prior to the change, roughly 13% of CPU usage for executing a join occurred within this method; this was reduced to approximately 8%. (Bug #95007, Bug #29640755)

Changes in MySQL NDB Cluster 8.0.22 (2020-10-19, General Availability)

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

Backup Notes

- To provide protection against unauthorized recovery of data from backups, this release adds support for `NDB` native encrypted backup using AES-256-CBC. Encrypted backup files are protected by a user-supplied password. `NDB` does not save this password; this needs to be done by the user or application. To create an encrypted backup, use `ENCRYPT PASSWORD=`*`password`* with the `ndb_mgm` client `START BACKUP` command (in addition to any other options which may be required). You can also initiate an encrypted backup in applications by calling the MGM API `ndb_mgm_start_backup4()` function.

  To restore from an encrypted backup, use `ndb_restore` with both of the options `--decrypt` and `--backup-password=`*`password`*. `ndb_print_backup_file` can also read encrypted files using the `-P` option added in this release.

  The encryption password used with this feature can be any string of up to 256 characters from the range of printable ASCII characters other than `!`, `'`, `"`, `$`, `%`, `\`, and `^`. When a password is supplied for encryption or decryption, it must be quoted using either single or double quotation marks. It is possible to specify an empty password using `''` or `""` but this is not recommended.

  You can encrypt existing backup files using the `ndbxfrm` utility which is added to the NDB Cluster distribution in this release; this program can also decrypt encrypted backup files. `ndbxfrm` also compresses and decompresses NDB Cluster backup files. The compression method is the same as used by NDB Cluster for creating compressed backups when `CompressedBackup` is enabled.

  It is also possible to require encrypted backups using `RequireEncryptedBackup`. When this parameter is enabled (by setting it equal to 1), the management client rejects any attempt to perform a backup that is not encrypted.

  For more information, see Using The NDB Cluster Management Client to Create a Backup, as well as ndbxfrm — Compress, Decompress, Encrypt, and Decrypt Files Created by NDB Cluster. (WL #13474, WL #13499, WL #13548)

Deprecation and Removal Notes

- **NDB Client Programs:** Effective with this release, the MySQL NDB Cluster Auto-Installer (`ndb_setup.py`) has been deprecated and is subject to removal in a future version of NDB Cluster. (Bug #31888835)
- **ndbmemcache:** `ndbmemcache` is deprecated in this release of NDB Cluster, and is scheduled for removal in the next release. (Bug #31876970)

Functionality Added or Changed

- **Important Change:** The `Ndb_metadata_blacklist_size` status variable was renamed as `Ndb_metadata_excluded_count`. (Bug #31465469)

- **Packaging:** Made the following improvements to the `server-minimal` RPM for NDB Cluster and the NDB Cluster Docker image:
  - Added `ndb_import` and other helpful utilities.
  - Included NDB utilities are now linked dynamically.
  - The NDB Cluster Auto-Installer is no longer included.
  - `ndbmemcache` is no longer included.

  (Bug #31838832)

- Added the CMake option `NDB_UTILS_LINK_DYNAMIC`, to allow dynamic linking of NDB utilities with `ndbclient`. (Bug #31668306)

- IPv6 addressing is now supported for connections to management and data nodes, including connections between management and data nodes with SQL nodes. For IPv6 addressing to work, the operating platform and network on which the cluster is deployed must support IPv6. Hostname resolution to IPv6 addresses must be provided by the operating platform (this is the same as when using IPv4 addressing).

  Mixing IPv4 and IPv6 addresses in the same cluster is not recommended, but this can be made to work in either of the following cases, provided that `--bind-address` is not used with `ndb_mgmd`:

  - Management node configured with IPv6, data nodes configured with IPv4: This works if the data nodes are started with `--ndb-connectstring` set to the IPv4 address of the management nodes.
  - Management node configured with IPv4, data nodes configured with IPv6: This works if the data nodes are started with `--ndb-connectstring` set to the IPv6 address of the management node.

  When upgrading from an NDB version that does not support IPv6 addressing to a version that does so, it is necessary that the network already support both IPv4 and IPv6. The software upgrade must be performed first; after this, you can update the IPv4 addresses used in the `config.ini` configuration file with the desired IPv6 addresses. Finally, in order for the configuration changes to take effect, perform a system restart of the cluster. (WL #12963)

Bugs Fixed

- **Important Change; NDB Cluster APIs:** The NDB Cluster adapter for Node.js was built against an obsolete version of the runtime. Now it is built using Node.js 12.18.3, and only that version or a later version of Node.js is supported by `NDB`. (Bug #31783049)

- **Important Change:** In order to synchronize excluded metadata objects, it was necessary to correct the underlying issue, if any, and then trigger the synchronization of the objects again. This could be achieved though discovery of individual tables, which does not scale well with an increase in the number of tables and SQL nodes. It could also be done by reconnecting the SQL node to the cluster, but doing so also incurs extra overhead.

  To fix this issue, the list of database objects excluded due to synchronization failure is cleared when `ndb_metadata_sync` is enabled by the user. This makes all such objects eligible for synchronization in the subsequent detection run, which simplifies retrying the synchronization of all excluded objects.

  This fix also removes the validation of objects to be retried which formerly took take place at the beginning of each detection run. Since these objects are of interest only while `ndb_metadata_sync` is enabled, the list of objects to be retried is cleared when this variable is disabled, signalling that all changes have been synchronized. (Bug #31569436)

- **Packaging:** The Dojo library included with NDB Cluster has been upgraded to version 1.15.4. (Bug #31559518)
- **NDB Disk Data:** `ndbmtd` sometimes terminated unexpectedly when it could not complete a lookup for a log file group during a restore operation. (Bug #31284086)
- **NDB Disk Data:** While upgrading a cluster having 3 or 4 replicas after creating sufficient disk data objects to fill up the tablespace, and while performing inserts on the disk data tables, trying to stop some data nodes caused others to exit improperly. (Bug #30922322)
- **NDB Cluster APIs:** In certain cases, the `Table::getColumn()` method returned the wrong `Column` object. This could happen when the full name of one table column was a prefix of the name of another, or when the names of two columns had the same hash value. (Bug #31774685)
- **NDB Cluster APIs:** It was possible to make invalid sequences of NDB API method calls using blobs. This was because some method calls implicitly cause transaction execution inline, to deal with blob parts and other issues, which could cause user-defined operations not to be handled correctly due to the use of a method executing operations relating to blobs while there still user-defined blob operations pending. Now in such cases, NDB raises a new error 4558 `Pending blob operations must be executed before this call`. (Bug #27772916)
- `ndb_restore --remap-column` did not handle columns containing `NULL` values correctly. Now any offset specified by the mapping function used with this option is not applied to `NULL`, so that `NULL` is preserved as expected. (Bug #31966676)
- The `ndb_print_backup_file` utility did not respect byte order for row data. This tool now performs byte swapping on row page information to ensure the same results on both big-endian and little-endian platforms. (Bug #31831438)

  References: See also: Bug #32470157.
- In some cases following an upgrade from a version of NDB Cluster previous to 8.0.18 to a later one, writing the `sysfile` (see NDB Cluster Data Node File System Directory) and reading back from it did not work correctly. This could occur when explicit node group assignments to data nodes had been made (using the `NodeGroup` parameter); it was possible for node group assignments to change spontaneously, and even possible for node groups not referenced in the configuration file to be added. This was due to issues with version 2 of the `sysfile` format introduced in NDB 8.0.18. (Bug #31828452, Bug #31820201)

  References: See also: Bug #31726653.
- After encountering the data node in the configuration file which used `NodeGroup=65536`, the management server stopped assigning data nodes lacking an explicit `NodeGroup` setting to node groups. (Bug #31825181)
- Data nodes in certain cases experienced fatal memory corruption in the `PGMAN` kernel block due to an invalid assumption that pages were 32KB aligned, when in fact they are normally aligned to the system page size (4096 or 8192 bytes, depending on platform). (Bug #31768450, Bug #31773234)
- Fixed a misspelled define introduced in NDB 8.0.20 which made an internal function used to control adaptive spinning non-operational. (Bug #31765660)
- When executing undo log records during undo log recovery it was possible when hitting a page cache miss to use the previous undo log record multiple times. (Bug #31750627)
- When an SQL node or cluster shutdown occurred during schema distribution while the coordinator was still waiting for the participants, the schema distribution was aborted halfway but any rows in `ndb_schema_result` related to this schema operation were not cleared. This left open the

possibility that these rows might conflict with a future reply from a participant if a DDL operation having the same schema operation ID originated from a client using the same node ID.

To keep this from happening, we now clear all such rows in `ndb_schema_result` during `NDB` binary log setup. This assures that there are no DDL distributions in progress and any rows remaining in the `ndb_schema_result` table are already obsolete. (Bug #31601674)

- Help output from the MySQL Cluster Auto-Installer displayed incorrect version information. (Bug #31589404)
- In certain rare circumstances, `NDB` missed checking for completion of a local checkpoint, leaving it uncompleted, which meant that subsequent local checkpoints could not be executed. (Bug #31577633)
- A data definition statement can sometimes involve reading or writing of multiple rows (or both) from tables; `NDBCLUSTER` starts an `NdbTransaction` to perform these operations. When such a statement was rolled back, `NDBCLUSTER` attempted to roll back the schema change before rolling back the `NdbTransaction` and closing it; this led to the rollback hanging indefinitely while the cluster waited for the `NdbTransaction` object to close before it was able to roll back the schema change.

  Now in such cases, `NDBCLUSTER` rolls back the schema change only after rolling back and closing any open Ndb`Transaction` associated with the change. (Bug #31546868)
- Adding a new user was not always synchronized correctly to all SQL nodes when the `NDB_STORED_USER` privilege was granted to the new user. (Bug #31486931)
- In some cases, `QMGR` returned conflicting `NDB` engine and MySQL server version information, which could lead to unplanned management node shutdown. (Bug #31471959)
- `SUMA` on a node that is starting up should not send a `DICT_UNLOCK_ORD` signal to the `DICT` block on the master node until both all `SUMA_HANDOVER_REQ` signals sent have had `SUMA_HANDOVER_CONF` signals sent in response, and every switchover bucket set up on receiving a `SUMA_HANDOVER_CONF` has completed switchover. In certain rare cases using `NoOfReplicas` > 2, and in which the delay between global checkpoints was unusually short, it was possible for some switchover buckets to be ready for handover before others, and for handover to proceed even though this was the case. (Bug #31459930)
- Attribute ID mapping needs to be performed when reading data from an `NDB` table using indexes or a primary key whose column order is different than that of the table. For unique indexes, a cached attribute ID map is created when the table is opened, and is then used for each subsequent read, but for primary key reads, the map was built for every read. This is changed so that an attribute ID map for primary key is built and cached when opening the table, and used whenever required for any subsequent reads. (Bug #31452597)

  References: See also: Bug #24444899.
- During different phases of the restore process, `ndb_restore` used different numbers of retries for temporary errors as well as different sleep times between retries. This is fixed by implementing consistent retry counts and sleep times across all restore phases. (Bug #31372923)
- Removed warnings generated when compiling `NDBCLUSTER` with Clang 10. (Bug #31344788)
- The `SPJ` block contains a load throttling mechanism used when generating `LQHKEYREQ` signals. When these were generated from parent rows from a scan, and this scan had a bushy topology with multiple children performing key lookups, it was possible to overload the job queues with too many `LQHKEYREQ` signals, causing node shutdowns due to full job buffers. This problem was originally fixed by Bug #14709490. Further investigation of this issue showed that `job buffer full` errors could occur even if the `SPJ` query was not bushy. Due to the increase in the internal batch size for `SPJ` workers in NDB 7.6.4 as part of work done to implement use of multiple fragments when

sending SCAN_FRAGREQ signals to the SPJ block, even a simple query could fill up the job buffers when a relatively small number of such queries were run in parallel.

To fix this problem, we no longer send any further LQHKEYREQ signals once the number of outstanding signals in a given request exceeds 256. Instead, the parent row from which the LQHKEYREQ is produced is buffered, and the correlation ID of this row is stored in the collection of operations to be resumed later. (Bug #31343524)

References: This issue is a regression of: Bug #14709490.

- MaxDiskWriteSpeedOwnRestart was not honored as an upper bound for local checkpoint writes during a node restart. (Bug #31337487)

  References: See also: Bug #29943227.

- Under certain rare circumstances, DROP TABLE of an NDB table triggered an assert. (Bug #31336431)

- During a node restart, the SUMA block of the node that is starting must get a copy of the subscriptions (events with subscribers) and subscribers (NdbEventOperation instances which are executing) from a node already running. Before the copy is complete, nodes which are still starting ignore any user-level SUB_START or SUB_STOP requests; after the copy is done, they can participate in such requests. While the copy operation is in progress, user-level SUB_START and SUB_STOP requests are blocked using a DICT lock.

  An issue was found whereby a starting node could participate in SUB_START and SUB_STOP requests after the lock was requested, but before it is granted, which resulted in unsuccessful SUB_START and SUB_STOP requests. This fix ensures that the nodes cannot participate in these requests until after the DICT lock has actually been granted. (Bug #31302657)

- Backups errored out with **FsErrInvalidParameters** when the filesystem was running with O_DIRECT and a data file write was not aligned with the 512-byte block size used by O_DIRECT writes. If the total fragment size in the data file is not aligned with the O_DIRECT block size, NDB pads the last write to the required size, but when there were no fragments to write, BACKUP wrote only the header and footer to the data file. Since the header and footer are less than 512 bytes, leading to the issue with the O_DIRECT write.

  This is fixed by padding out the generic footer to 512 bytes if necessary, using an EMPTY_ENTRY, when closing the data file. (Bug #31180508)

- When employing an execution strategy which requires it to buffer received key rows for later use, DBSPJ now manages the buffer memory allocation tree node by tree node, resulting in a significant drop in CPU usage by the DBSPJ block. (Bug #31174015)

- DBSPJ now uses linear memory instead of segmented memory for storing and handling TRANSID_AI signals, which saves approximately 10% of the CPU previously consumed. Due to this change, it is now possible for DBSPJ to accept TRANSID_AI signals in the short signal format; this is more efficient than the long signal format which requires segmented memory. (Bug #31173582, Bug #31173766)

- Altering the table comment of a fully replicated table using ALGORITHM=INPLACE led to an assertion. (Bug #31139313)

- A local data manager (LDM) has a mechanism for ensuring that a fragment scan does not continue indefinitely when it finds too few rows to fill the available batch size in a reasonable amount of time (such as when a ScanFilter evaluates to false for most of the scanned rows). When this time limit, set in DBLQH as 10 ms, has expired, any rows found up to that point are returned, independent

of whether the specified batch size has been filled or not. This acts as a keep-alive mechanism between data and API nodes, as well as to avoid keeping any locks held during the scan for too long.

A side effect of this is that returning result row batches to the DBSPJ block which are filled well below the expected limit could cause performance issues. This was due not only to poor utilization of the space reserved for batches, requiring more NEXTREQ round trips, but because it also caused DBSPJ internal parallelism statistics to become unreliable.

Since the DBSPJ block never requests locks when performing scans, overly long locks are not a problem for SPJ requests. Thus it is considered safe to let scans requested by DBSPJ to continue for longer than the 10 ms allowed previously, and the limit set in DBLQH has been increased to 100 ms. (Bug #31124065)

- For a pushed join, the output from EXPLAIN FORMAT=TREE did not indicate whether the table access was an index range scan returning multiple rows, or a single-row lookup on a primary or unique key.

  This fix provides also a minor optimization, such that the handler interface is not accessed more than once in an attempt to return more than a single row if the access type is known to be Unique. (Bug #31123930)

- A previous change (made in NDB 8.0.20) made it possible for a pushed join on tables allowing READ_BACKUP to place two SPJ workers on the data node local to the DBTC block while placing no SPJ workers on some other node; this sometime imbalance is intentional, as the SPJ workload (and possible introduced imbalance) is normally quite low compared to the gains of enabling more local reads of the backup fragments. As an unintended side effect of the same change, these two colocated SPJ workers might scan the same subset of fragments in parallel; this broke an assumption in the DBSPJ block that only a single SPJ worker is instantiated on each data node on which the logic for insuring that each SPJ worker starts its scans from a different fragment depends.

  To fix this problem, the starting fragment for each SPJ worker is now calculated based on the root fragment ID from which the worker starts, which is unique among all SPJ workers even when some of them reside on the same node. (Bug #31113005)

  References: See also: Bug #30639165.

- When upgrading a cluster from NDB 8.0.17 or earlier to 8.0.18 or later, data nodes not yet upgraded could shut down unexpectedly following upgrade of the management server (or management servers) to the new software version. This occurred when a management client STOP command was sent to one or more of the data nodes still running the old version and the new master node (also running the old version of the NDB software) subsequently underwent an unplanned shutdown.

  It was found that this occurred due to setting the signal length and number of signal sections incorrectly when sending a GSN_STOP_REQ—one of a number of signals whose length has been increased in NDB 8.0 as part of work done to support greater numbers of data nodes—to the new master. This happened due to the use of stale data retained from sending a GSN_STOP_REQ to the previous master node. To prevent this from happening, ndb_mgmd now sets the signal length and number of sections explicitly each time, prior to sending a GSN_STOP_REQ signal. (Bug #31019990)

- In some cases, when failures occurred while replaying logs and restoring tuples, ndb_restore terminated instead of returning an error. In addition, the number of retries to be attempted for some operations was determined by hard-coded values. (Bug #30928114)

- During schema distribution, if the client was killed after a DDL operation was already logged in the ndb_schema table, but before the participants could reply, the client simply marked all participants as failed in the NDB_SCHEMA_OBJECT and returned. Since the distribution protocol was already in progress, the coordinator continued to wait for the participants, received their ndb_schema_result insert and processed them; meanwhile, the client was open to send another DDL operation; if one was executed and distribution of it was begun before the coordinator could finish processing the

previous schema change, this triggered an assertion there should be only one distribution of a schema operation active at any given time.

In addition, when the client returned having detected a thread being killed, it also released the global schema lock (GSL); this could also lead to undefined issues since the participant could make the changes under the assumption that the GSL was still being held by the coordinator.

In such cases, the client should not return after the DDL operation has been logged in the `ndb_schema` table; from this point, the coordinator has control and the client should wait for it to make a decision. Now the coordinator aborts the distribution only in the event of a server or cluster shutdown, and otherwise waits for all participants either to reply, or to time out and mark the schema operation as completed. (Bug #30684839)

- When, during a restart, a data node received a `GCP_SAVEREQ` signal prior to beginning start phase 9, and thus needed to perform a global checkpoint index write to a local data manager's local checkpoint control file, it did not record information from the `DIH` block originating with the node that sent the signal as part of the data written. This meant that, later in start phase 9, when attempting to send a `GCP_SAVECONF` signal in response to the `GCP_SAVEREQ`, this information was not available, which meant the response could not be sent, resulting in an unplanned shutdown of the data node. (Bug #30187949)

- Setting `EnableRedoControl` to `false` did not fully disable `MaxDiskWriteSpeed`, `MaxDiskWriteSpeedOtherNodeRestart`, and `MaxDiskWriteSpeedOwnRestart` as expected. (Bug #29943227)

  References: See also: Bug #31337487.

- A `BLOB` value is stored by `NDB` in multiple parts; when reading such a value, one read operation is executed per part. If a part is not found, the read fails with a `row not found error`, which indicates a corrupted `BLOB`, since a `BLOB` should never have any missing parts. A problem can arise because this error is reported as the overall result of the read operation, which means that mysqld sees no error and reports zero rows returned.

  This issue is fixed by adding a check specifically for the case in wich a blob part is not found. Now, when this occurs, overwriting the `row not found` error with `corrupted blob`, which causes the originating `SELECT` statement to fail as expected. Users of the NDB API should be aware that, despite this change, the `NdbBlob::getValue()` method continues to report the error as `row not found` in such cases. (Bug #28590428)

- Data nodes did not start when the `RealtimeScheduler` configuration parameter was set to 1. This was due to the fact that index builds during startup are performed by temporarily diverting some I/O threads for use as index building threads, and these threads inherited the realtime properties of the I/O threads. This caused a conflict (treated as a fatal error) when index build thread specifications were checked to ensure that they were not realtime threads. This is fixed by making sure that index build threads are not treated as realtime threads regardless of any settings applying to their host I/O threads, which is as actually intended in their design. (Bug #27533538)

- Using an in-place `ALTER TABLE` to drop an index could lead to the unplanned shutdown of an SQL node. (Bug #24444899)

- As the final step when executing `ALTER TABLE ... ALGORITHM=INPLACE`, `NDBCLUSTER` performed a read of the table metadata from the `NDB` dictionary, requiring an extra round trip between the SQL nodes and data nodes, which unnecessarily both slowed down execution of the statement and provided an avenue for errors which `NDBCLUSTER` was not prepared to handle correctly. This issue is fixed by removing the read of `NDB` table metadata during the final phase of executing an in-place `ALTER TABLE` statement. (Bug #99898, Bug #31497026)

- A memory leak could occur when preparing an `NDB` table for an in-place `ALTER TABLE`. (Bug #99739, Bug #31419144)

- Added the `AllowUnresolvedHostNames` configuration parameter. When set to `true`, this parameter overrides the fatal error normally raised when `ndb_mgmd` cannot connect to a given host name, allowing startup to continue and generating only a warning instead. To be effective, the parameter must be set in the cluster global configuration file's `[tcp default]` section. (WL #13860)

Changes in MySQL NDB Cluster 8.0.21 (2020-07-13, General Availability)

- Packaging Notes
- Functionality Added or Changed
- Bugs Fixed

Packaging Notes

- For Windows, MSI installer packages for NDB Cluster now include a check for the required Visual Studio redistributable package, and produce a message asking the user to install it if it is missing. (Bug #30541398)

Functionality Added or Changed

- **NDB Disk Data:** An initial restart of the cluster now causes the removal of all `NDB` tablespaces and log file groups from the `NDB` dictionary and the MySQL data dictionary. This includes the removal of all data files and undo log files associated with these objects. (Bug #30435378)

  References: See also: Bug #29894166.

- The status variable `Ndb_metadata_blacklist_size` is now deprecated, and is replaced in NDB 8.0.22 by `Ndb_metadata_excluded_count`. (Bug #31465469)

- It now possible to consolidate data from separate instances of NDB Cluster into a single target NDB Cluster when the original datasets all use the same schema. This is supported when using backups created using `START BACKUP` in `ndb_mgm` and restoring them with `ndb_restore`, using the `--remap-column` option implemented in this release (along with `--restore-data` and possibly other options). `--remap-column` can be employed to handle cases of overlapping primary, unique, or both sorts of key values between source clusters, and you need to make sure that they do not overlap in the target cluster. This can also be done to preserve other relationships between tables.

  When used together with `--restore-data`, the new option applies a function to the value of the indicated column. The value set for this option is a string of the format *`db.tbl.col:fn:args`*, whose components are listed here:

  - *`db`*: Database name, after performing any renames.
  - *`tbl`*: Table name.
  - *`col`*: Name of the column to be updated. This column's type must be one of `INT` or `BIGINT`, and can optionally be `UNSIGNED`.
  - *`fn`*: Function name; currently, the only supported name is `offset`.
  - *`args`*: The size of the offset to be added to the column value by `offset`. The range of the argument is that of the signed variant of the column's type; thus, negative offsets are supported.

  You can use `--remap-column` for updating multiple columns of the same table and different columns of different tables, as well as combinations of multiple tables and columns. Different offset values can be employed for different columns of the same table.

  As part of this work, two new options are also added to `ndb_desc` in this release:

- `--auto-inc` (short form `-a`): Includes the next auto-increment value in the output, if the table has an `AUTO_INCREMENT` column.
- `--context` (short form `-x`): Provides extra information about the table, including the schema, database name, table name, and internal ID.

These options may be useful for obtaining information about `NDB` tables when planning a merge, particularly in situations where the `mysql` client may not be readily available.

For more information, see the descriptions for `--remap-column`, `--auto-inc`, and `--context`. (Bug #30383950, WL #11796)

- Detailed real-time information about the state of automatic metadata mismatch detection and synchronization can now be obtained from tables in the MySQL Performance Schema. These two tables are listed here:
  - `ndb_sync_pending_objects`: Contains information about `NDB` database objects for which mismatches have been detected between the `NDB` dictionary and the MySQL data dictionary. It does not include objects which have been excluded from mismatch detection due to permanent errors raised when attempting to synchronize them.
  - `ndb_sync_excluded_objects`: Contains information about `NDB` database objects which have been excluded because they cannot be synchronized between the `NDB` dictionary and the MySQL data dictionary, and thus require manual intervention. These objects are no longer subject to mismatch detection until such intervention has been performed.

  In each of these tables, each row corresponds to a database object, and contains the database object's parent schema (if any), the object's name, and the object's type. Types of objects include schemas, tablespaces, log file groups, and tables. The `ndb_sync_excluded_objects` table shows in addition to this information the reason for which the object has been excluded.

  Performance Schema NDB Cluster Tables, provides further information about these Performance Schema tables. (Bug #30107543, WL #13712)

- `ndb_restore` now supports different primary key definitions for source and target tables when restoring from an `NDB` native backup, using the `--allow-pk-changes` option introduced in this release. Both increasing and decreasing the number of columns making up the original primary key are supported. This may be useful when it is necessary to accommodate schema version changes while restoring data, or when doing so is more efficient or less time-consuming than performing `ALTER TABLE` statements involving primary key changes on a great many tables following the restore operation.

  When extending a primary key with additional columns, any columns added must not be nullable, and any values stored in any such columns must not change while the backup is being taken. Changes in the values of any such column while trying to add it to the table's primary key causes the restore operation to fail. Due to the fact that some applications set the values of all columns when updating a row even if the values of one or more of the columns does not change, it is possible to override this behavior by using the `--ignore-extended-pk-updates` option which is also added in this release. If you do this, care must be taken to insure that such column values do not actually change.

  When removing columns from the table's primary key, it is not necessary that the columns dropped from the primary key remain part of the table afterwards.

  For more information, see the description of the `--allow-pk-changes` option in the documentation for `ndb_restore`. (Bug #26435136, Bug #30383947, Bug #30634010, WL #10730)

- Added the `--ndb-log-fail-terminate` option for `mysqld`. When used, this causes the SQL node to terminate if it is unable to log all row events. (Bug #21911930)

  References: See also: Bug #30383919.

- When a scalar subquery has no outer references to the table to which the embedding condition is attached, the subquery may be evaluated independent of that table; that is, the subquery is not dependent. `NDB` now attempts to identify and evaluate such a subquery before trying to retrieve any rows from the table to which it is attached, and to use the value thus obtained in a pushed condition, rather than using the subquery which provided the value. (WL #13798)

- In MySQL 8.0.17 and later, the MySQL Optimizer transforms `NOT EXISTS` and `NOT IN` queries into antijoins. `NDB` can now push these down to the data nodes.

  This can be done when there is no unpushed condition on the table, and the query fulfills any other conditions which must be met for an outer join to be pushed down. (WL #13796, WL #13978)

Bugs Fixed

- **Important Change; NDB Disk Data:** An online change of tablespace is not supported for `NDB` tables. Now, for an `NDB` table, the statement `ALTER TABLE ndb_table ... ALGORITHM=INPLACE, TABLESPACE=new_tablespace` is specifically disallowed.

  As part of this fix, the output of the `ndb_desc` utility is improved to include the tablespace name and ID for an `NDB` table which is using one. (Bug #31180526)

- The wrong index was used in the array of indexes while dropping an index. For a table with 64 indexes this caused uninitialized memory to be released. This problem also caused a memory leak when a new index was created at any later time following the drop. (Bug #31408095)

- Removed an unnecessary dependency of `ndb_restore` on the `NDBCLUSTER` plugin. (Bug #31347684)

- Objects for which auto-synchronization fails due to temporary errors, such as failed acquisitions of metadata locks, are simply removed from the list of detected objects, making such objects eligible for detection in later cycles in which the synchronization is retried and hopefully succeeds. This best-effort approach is suitable for the default auto-synchronization behaviour but is not ideal when the using the `ndb_metadata_sync` system variable, which triggers synchronization of all metadata, and when synchronization is complete, is automatically set to false to indicate that this has been done.

  What happened, when a temporary error persisted for a sizable length of time, was that metadata synchronization could take much longer than expected and, in extreme cases, could hang indefinitely, pending user action. One such case occurred when using `ndb_restore` with the `--disable-indexes` option to restore metadata, when the synchronization process entered a vicious cycle of detection and failed synchronization attempts due to the missing indexes until the indexes were rebuilt using `ndb_restore --rebuild-indexes`.

  The fix for this issue is, whenever `ndb_metadata_sync` is set to `true`, to exclude an object after synchronization of it fails 10 times with temporary errors by promoting these errors to a permanent error, in order to prevent stalling. This is done by maintaining a list of such objects, this list including a count of the number of times each such object has been retried. Validation of this list is performed during change detection in a similar manner to validation of the exclusion list. (Bug #31341888)

- 32-bit platforms are not supported by NDB 8.0. Beginning with this release, the build process checks the system architecture and aborts if it is not 64-bit. (Bug #31340969)

- Page-oriented allocations on the data nodes are divided into nine resource groups, some having pages dedicated to themselves, and some having pages dedicated to shared global memory which can be allocated by any resource group. To prevent the query memory resource group from depriving other, more important resource groups such as transaction memory of resources,

allocations for query memory are performed with low priority and are not allowed to use the last 10% of shared global memory. This change was introduced by poolification work done in NDB 8.0.15.

Subsequently, it was observed that the calculation for the number of pages of shared global memory kept inaccessible to query memory was correct only when no pages were in use, which is the case when the `LateAlloc` data node parameter is disabled (0).

This fix corrects that calculation as performed when `LateAlloc` is enabled. (Bug #31328947)

References: See also: Bug #31231286.

- Multi-threaded restore is able to drive greater cluster load than the previous single-threaded restore, especially while restoring of the data file. To avoid load-related issues, the insert operation parallelism specified for an `ndb_restore` instance is divided equally among the part threads, so that a multithreaded instance has a similar level of parallelism for transactions and operations to a single-threaded instance.

  An error in division caused some part threads to have lower insert operation parallelism than they should have, leading to an slower restore than expected. This fix ensures all part threads in a multi-threaded `ndb_restore` instance get an equal share for parallelism. (Bug #31256989)

- `DUMP 1001` (`DumpPageMemoryOnFail`) now prints out information about the internal state of the data node page memory manager when allocation of pages fails due to resource constraints. (Bug #31231286)

- Statistics generated by `NDB` for use in tracking internal objects allocated and deciding when to release them were not calculated correctly, with the result that the threshold for resource usage was 50% higher than intended. This fix corrects the issue, and should allow for reduced memory usage. (Bug #31127237)

- The Dojo toolkit included with NDB Cluster and used by the Auto-Installer was upgraded to version 1.15.3. (Bug #31029110)

- A packed version 1 configuration file returned by `ndb_mgmd` could contain duplicate entries following an upgrade to NDB 8.0, which made the file incompatible with clients using version 1. This occurs due to the fact that the code for handling backwards compatibility assumed that the entries in each section were already sorted when merging it with the default section. To fix this, we now make sure that this sort is performed prior to merging. (Bug #31020183)

- When executing any of the `SHUTDOWN`, `ALL STOP`, or `ALL RESTART` management commands, it is possible for different nodes to attempt to stop on different global checkpoint index (CGI) boundaries. If they succeed in doing so, then a subsequent system restart is slower than normal because any nodes having an earlier stop GCI must undergo takeover as part of the process. When nodes failing on the first GCI boundary cause surviving nodes to be nonviable, surviving nodes suffer an arbitration failure; this has the positive effect of causing such nodes to halt at the correct GCI, but can give rise to spurious errors or similar.

  To avoid such issues, extra synchronization is now performed during a planned shutdown to reduce the likelihood that different data nodes attempt to shut down at different GCIs as well as the use of unnecessary node takeovers during system restarts. (Bug #31008713)

- During an upgrade, a client could connect to an NDB 8.0 data node without specifying a multiple transporter instance ID, so that this ID defaulted to -1. Due to an assumption that this would occur only in the Node starting state with a single transporter, the node could hang during the restart. (Bug #30899046)

- When an NDB cluster was upgraded from a version that does not support the data dictionary to one that does, any DDL executed on a newer SQL node was not properly distributed to older ones. In addition, newer SDI generated during DDL execution was ignored by any data nodes that had not yet been upgraded. These two issues led to schema states that were not consistent between nodes of different NDB software versions.

We fix this problem by blocking any DDL affecting NDB data objects while an upgrade from a previous NDB version to a version with data dictionary support is ongoing. (Bug #30877440)

References: See also: Bug #30184658.

- The `mysql.ndb_schema` table, used internally for schema distribution among SQL nodes, has been modified in NDB 8.0. When a cluster is being upgraded from a older version of NDB, the first SQL node to be upgraded updates the definition of this table to match that used by NDB 8.0 GA releases. (For this purpose, NDB now uses 8.0.21 as the cutoff version.) This is done by dropping the existing table and re-creating it using the newer definition. SQL nodes which have not yet been upgraded receive this `ndb_schema` table drop event and enter read-only mode, becoming writable again only after they are upgraded.

  To keep SQL nodes running older versions of NDB from going into read-only mode, we change the upgrade behavior of `mysqld` such that the `ndb_schema` table definition is updated only if all SQL nodes connected to the cluster are running an 8.0 GA version of NDB and thus having the updated `ndb_schema` table definition. This means that, during an upgrade to the current or any later version, no MySQL Server that is being upgraded updates the `ndb_schema` table if there is at least one SQL node with an older version connected to the cluster. Any SQL node running an older version of NDB remains writable throughout the upgrade process. (Bug #30876990, Bug #31016905)

- `ndb_import` did not handle correctly the case where a CSV parser error occurred in a block of input other than the final block. (Bug #30839144)

- When `mysqld` was upgraded to a version that used a new SDI version, all `NDB` tables become inaccessible. This was because, during an upgrade, synchronization of `NDB` tables relies on deserializing the SDI packed into the NDB Dictionary; if the SDI format was of an version older than that used prior to the upgrade, deserialization could not take place if the format was not the same as that of the new version, which made it impossible to create a table object in the MySQL data dictionary.

  This is fixed by making it possible for `NDB` to bypass the SDI version check in the MySQL server when necessary to perform deserialization as part of an upgrade. (Bug #30789293, Bug #30825260)

- When responding to a `SCANTABREQ`, an API node can provide a distribution key if it knows that the scan should work on only one fragment, in which case the distribution key should be the fragment ID, but in some cases a hash of the partition key was used instead, leading to failures in `DBTC`. (Bug #30774226)

- Several memory leaks found in `ndb_import` have been removed. (Bug #30756434, Bug #30727956)

- The master node in a backup shut down unexpectedly on receiving duplicate replies to a `DEFINE_BACKUP_REQ` signal. These occurred when a data node other than the master errored out during the backup, and the backup master handled the situation by sending itself a `DEFINE_BACKUP_REF` signal on behalf of the missing node, which resulted in two replies being received from the same node (a `CONF` signal from the problem node prior to shutting down and the `REF` signal from the master on behalf of this node), even though the master expected only one reply per node. This scenario was also encountered for `START_BACKUP_REQ` and `STOP_BACKUP_REQ` signals.

  This is fixed in such cases by allowing duplicate replies when the error is the result of an unplanned node shutdown. (Bug #30589827)

- When updating `NDB_TABLE` comment options using `ALTER TABLE`, other options which has been set to non-default values when the table was created but which were not specified in the `ALTER TABLE` statement could be reset to their defaults.

  See Setting NDB Comment Options, for more information. (Bug #30428829)

- Removed a memory leak found in the `ndb_import` utility. (Bug #29820879)
- Incorrect handling of operations on fragment replicas during node restarts could result in a forced shutdown, or in content diverging between fragment replicas, when primary keys with nonbinary (case-sensitive) equality conditions were used. (Bug #98526, Bug #30884622)

Changes in MySQL NDB Cluster 8.0.20 (2020-04-27, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change:** It is now possible to divide a backup into slices and to restore these in parallel using two new options implemented for the `ndb_restore` utility, making it possible to employ multiple instances of `ndb_restore` to restore subsets of roughly the same size of the backup in parallel, which should help to reduce the length of time required to restore an NDB Cluster from backup.

  The `--num-slices` options determines the number of slices into which the backup should be divided; `--slice-id` provides the ID of the slice (0 to 1 less than the number of slices) to be restored by `ndb_restore`.

  Up to 1024 slices are supported.

  For more information, see the descriptions of the `--num-slices` and `--slice-id` options. (Bug #30383937, WL #10691)

- **Important Change:** To increase the rate at which update operations can be processed, `NDB` now supports and by default makes use of multiple transporters per node group. By default, the number of transporters used by each node group in the cluster is equal to the number of the number of local data management (LDM) threads. While this number should be optimal for most use cases, it can be adjusted by setting the value of the `NodeGroupTransporters` data node configuration parameter which is introduced in this release. The maximum is the greater of the number of LDM threads or the number of TC threads, up to an overall maximum of 32 transporters.

  See Multiple Transporters, for additional information. (WL #12837)

- **NDB Client Programs:** Two options are added for the `ndb_blob_tool` utility, to enable it to detect missing blob parts for which inline parts exist, and to replace these with placeholder blob parts (consisting of space characters) of the correct length. To check whether there are missing blob parts, use the `ndb_blob_tool --check-missing` option. To replace with placeholders any blob parts which are missing, use the program's `--add-missing` option, also added in this release. (Bug #28583971)

- **NDB Client Programs:** Removed a dependency from the `ndb_waiter` and `ndb_show_tables` utility programs on the `NDBT` library. This library, used in `NDB` development for testing, is not required for normal use. The visible effect for users from this change is that these programs no longer print `NDBT_ProgramExit - status` following completion of a run. Applications that depend upon this behavior should be updated to reflect this change when upgrading to this release. (WL #13727, WL #13728)

- **MySQL NDB ClusterJ:** The unused `antlr3` plugin has been removed from the ClusterJ `pom` file. (Bug #29931625)

- **MySQL NDB ClusterJ:** The minimum Java version ClusterJ supports for MySQL NDB Cluster 8.0 is now Java 8. (Bug #29931625)

- **MySQL NDB ClusterJ:** A few Java APIs used by ClusterJ are now deprecated in recent Java versions. These adjustments have been made to ClusterJ:

- Replaced all `Class.newInstance()` calls with `Class.getDeclaredConstructor().newInstance()` calls. Also updated the exception handling and the test cases wherever required.
- All the `Number` classes' constructors that instantiate an object from a `String` or a primitive type are deprecated. Replaced all such deprecated instantiation calls with the corresponding `valueOf()` method calls.
- The `Proxy.getProxyClass()` is now deprecated. The `DomainTypeHandlerImpl` class now directly creates a new instance using the `Proxy.newProxyInstance()` method; all references to the `Proxy` class and its constructors are removed from the `DomainTypeHandlerImpl` class. `SessionFactoryImpl` class now uses the interfaces underlying the proxy object to identify the domain class rather than using the Proxy class. Also updated `DomainTypeHandlerFactoryTest`.
- The `finalize()` method is now deprecated. This patch does not change the overriding finalize() methods, but just suppresses the warnings on them. This deprecation will be handled separately in a later patch.
- Updated the CMake configuration to treat deprecation warnings as errors when compiling ClusterJ.

(Bug #29931625)

- `NDB` now supports versioning for `ndbinfo` tables, and maintains the current definitions for its tables internally. At startup, `NDB` compares its supported `ndbinfo` version with the version stored in the data dictionary. If the versions differ, `NDB` drops any old `ndbinfo` tables and recreates them using the current definitions. (WL #11563)
- Many outer joins and semijoins which previously could not be pushed down to the data nodes can now pushed (see Engine Condition Pushdown Optimization).

  Outer joins which can now be pushed include those which meet the following conditions:

  - There are no unpushed conditions on this table
  - There are no unpushed conditions on other tables in the same join nest, or in upper join nests on which it depends
  - All other tables in the same join nest, or in upper join nests on which it depends are also pushed

  A semijoin using an index scan can now be pushed if it meets the the conditions just noted for a pushed outer join, and it uses the `firstMatch` strategy. (WL #7636, WL #13576)

  References: See also: Bug #28728603, Bug #28672214, Bug #29296615, Bug #29232744, Bug #29161281, Bug #28728007.

- A new and simplified interface is implemented for enabling and configuring adaptive CPU spin. The `SpinMethod` data node parameter, added in this release, provides the following four settings:

  - `StaticSpinning`: Disables adaptive spinning; uses the static spinning employed in previous NDB Cluster releases
  - `CostBasedSpinning`: Enables adaptive spinning using a cost-based model
  - `LatencyOptimisedSpinning`: Enables adaptive spinning optimized for latency
  - `DatabaseMachineSpinning`: Enables adaptive spinning optimized for machines hosting databases, where each thread has its own CPU

  Each of these settings causes the data node to use a set of predetermined values, as needed, for one or more of the spin parameters listed here:

  - `SchedulerSpinTimer`: The data node configuration parameter of this name.
  - `EnableAdaptiveSpinning`: Enables or disables adaptive spinning; cannot be set directly in the cluster configuration file, but can be controlled directly using `DUMP 104004`
  - `SetAllowedSpinOverhead`: CPU time to allow to gain latency; cannot be set directly in the `config.ini` file, but possible to change directly, using `DUMP 104002`

  The presets available from `SpinMethod` should cover most use cases, but you can fine-tune the adaptive spin behavior using the `SchedulerSpinTimer` data node configuration parameter and the `DUMP` commands just listed, as well as additional `DUMP` commands in the `ndb_mgm` cluster management client; see the description of `SchedulerSpinTimer` for a complete listing.

  NDB 8.0.20 also adds a new TCP configuration parameter `TcpSpinTime` which sets the time to spin for a given TCP connection. This can be used to enable adaptive spinning for any such connections between data nodes, management nodes, and SQL or API nodes.

  The `ndb_top` tool is also enhanced to provide spin time information per thread; this is displayed in green in the terminal window.

  For more information, see the descriptions of the `SpinMethod` and `TcpSpinTime` configuration parameters, the `DUMP` commands listed or indicated previously, and the documentation for `ndb_top`. (WL #12554)

Bugs Fixed

- **Important Change:** When `lower_case_table_names` was set to 0, issuing a query in which the lettercase of any foreign key names differed from the case with which they were created led to an unplanned shutdown of the cluster. This was due to the fact that mysqld treats foreign key names as case insensitive, even on case-sensitive file systems, whereas the manner in which the `NDB` dictionary stored foreign key names depended on the value of `lower_case_table_names`, such that, when this was set to 0, during lookup, `NDB` expected the lettercase of any foreign key names to match that with which they were created. Foreign key names which differed in lettercase could then not be found in the `NDB` dictionary, even though it could be found in the MySQL data dictionary, leading to the previously described issue in `NDBCLUSTER`.

  This issue did not happen when `lower_case_table_names` was set to 1 or 2.

  The problem is fixed by making foreign key names case insensitive and removing the dependency on `lower_case_table_names`. This means that the following two items are now always true:

  1. Foreign key names are now stored using the same lettercase with which they are created, without regard to the value of `lower_case_table_names`.
  2. Lookups for foreign key names by `NDB` are now always case insensitive.

(Bug #30512043)

- **Packaging:** Removed an unnecessary dependency on Perl from the `mysql-cluster-community-server-minimal` RPM package. (Bug #30677589)
- **Packaging:** `NDB` did not compile successfully on Ubuntu 16.04 with GCC 5.4 due to the use of `isnan()` rather than `std::isnan()`. (Bug #30396292)

  References: This issue is a regression of: Bug #30338980.
- **OS X:** Removed the variable `SCHEMA_UUID_VALUE_LENGTH` which was used only once in the `NDB` sources, and which caused compilation warnings when building on Mac OSX. The variable has been replaced with `UUID_LENGTH`. (Bug #30622139)
- **NDB Disk Data:** Allocation of extents in tablespace data files is now performed in round-robin fashion among all data files used by the tablespace. This should provide more even distribution of data in cases where multiple storage devices are used for Disk Data storage. (Bug #30739018)
- **NDB Disk Data:** Under certain conditions, checkpointing of Disk Data tables could not be completed, leading to an unplanned data node shutdown. (Bug #30728270)
- **NDB Disk Data:** An uninitialized variable led to issues when performing Disk Data DDL operations following a restart of the cluster. (Bug #30592528)
- **MySQL NDB ClusterJ:** When a `Date` value was read from a NDB cluster, ClusterJ sometimes extracted the wrong year value from the row. It was because the `Utility` class, when unpacking the `Date` value, wrongly extracted some extra bits for the year. This patch makes ClusterJ only extract the required bits. (Bug #30600320)
- **MySQL NDB ClusterJ:** When the cluster's `NdbOperation::AbortOption` type had the value of `AO_IgnoreOnError`, when there was a read error, ClusterJ took that as the row was missing and returned `null` instead of an exception. This was because with `AO_IgnoreOnErro`, the `execute()` method always returns a success code after each transaction, and ClusterJ is supposed to check for any errors in any of the individual operations; however, read operations were not checked by ClusterJ in the case. With this patch, read operations are now checked for errors after query executions, so that a reading error is reported as such. (Bug #30076276)
- The fix for a previous issue in the MySQL Optimizer adversely affected engine condition pushdown for the `NDB` storage engine. (Bug #303756135)

  References: This issue is a regression of: Bug #97552, Bug #30520749.
- When restoring signed auto-increment columns, `ndb_restore` incorrectly handled negative values when determining the maximum value included in the data. (Bug #30928710)
- On an SQL node which had been started with `--ndbcluster`, before any other nodes in the cluster were started, table creation succeeded while creating the `ndbinfo` schema, but creation of views did not, raising `HA_ERR_NO_CONNECTION` instead. (Bug #30846678)
- Formerly (prior to NDB 7.6.4) an `SPJ` worker instance was activated for each fragment of the root table of the pushed join, but in NDB 7.6 and later, a single worker is activated for each data node and is responsible for all fragments on that data node.

  Before this change was made, it was sufficient for each such worker to scan a fragment with parallelism equal to 1 for all `SPJ` workers to keep all local data manager threads busy. When the number of workers was reduced as result of the change, the minimum parallelism should have been increased to equal the number of fragments per worker to maintain the degree of parallelism.

  This fix ensures that this is now done. (Bug #30639503)
- The `ndb_metadata_sync` system variable is set to true to trigger synchronization of metadata between the MySQL data dictionary and the `NDB` dictionary; when synchronization is complete, the

variable is automatically reset to false to indicate that this has been done. One scenario involving the detection of a schema not present in the MySQL data dictionary but in use by the NDB Dictionary sometimes led to ndb_metadata_sync being reset before all tables belonging to this schema were successfully synchronized. (Bug #30627292)

- When using shared user and grants, all ALTER USER statements were distributed as snapshots, whether they contained plaintext passwords or not.

  In addition, SHOW CREATE USER did not include resource limits (such as MAX_QUERIES_PER_HOUR) that were set to zero, which meant that these were not distributed among SQL nodes. (Bug #30600321)

- Two buffers used for logging in QMGR were of insufficient size. (Bug #30598737)

  References: See also: Bug #30593511.

- Removed extraneous debugging output relating to SPJ from the node out logs. (Bug #30572315)

- When performing an initial restart of an NDB Cluster, each MySQL Server attached to it as an SQL node recognizes the restart, reinstalls the ndb_schema table from the data dictionary, and then clears all NDB schema definitions created prior to the restart. Because the data dictionary was cleared only after ndb_schema is reinstalled, installation sometimes failed due to ndb_schema having the same table ID as one of the tables from before the restart was performed. This issue is fixed by ensuring that the data dictionary is cleared before the ndb_schema table is reinstalled. (Bug #30488610)

- NDB sometimes made the assumption that the list of nodes containing index statistics was ordered, but this list is not always ordered in the same way on all nodes. This meant that in some cases NDB ignored a request to update index statistics, which could result in stale data in the index statistics tables. (Bug #30444982)

- When the optimizer decides to presort a table into a temporary table, before later tables are joined, the table to be sorted should not be part of a pushed join. Although logic was present in the abstract query plan interface to detect such query plans, that this did not detect correctly all situations using filesort into temporary table. This is changed to check whether a filesort descriptor has been set up; if so, the table content is sorted into a temporary file as its first step of accessing the table, which greatly simplifies interpretation of the structure of the join. We now also detect when the table to be sorted is a part of a pushed join, which should prevent future regressions in this interface. (Bug #30338585)

- When a node ID allocation request failed with **NotMaster** temporary errors, the node ID allocation was always retried immediately, without regard to the cause of the error. This caused a very high rate of retries, whose effects could be observed as an excessive number of Alloc node id for node *nnn* failed log messages (on the order of 15,000 messages per second). (Bug #30293495)

- For NDB tables having no explicit primary key, NdbReceiverBuffer could be allocated with too small a size. This was due to the fact that the attribute bitmap sent to NDB from the data nodes always includes the primary key. The extra space required for hidden primary keys is now taken into consideration in such cases. (Bug #30183466)

- When translating an NDB table created using .frm files in a previous version of NDB Cluster and storing it as a table object in the MySQL data dictionary, it was possible for the table object to be committed even when a mismatch had been detected between the table indexes in the MySQL data dictionary and those for the same table's representation the NDB dictionary. This issue did not occur for tables created in NDB 8.0, where it is not necessary to upgrade the table metadata in this fashion.

  This problem is fixed by making sure that all such comparisons are actually performed before the table object is committed, regardless of whether the originating table was created with or without the use of .frm files to store its metadata. (Bug #29783638)

- An error raised when obtaining cluster metadata caused a memory leak. (Bug #97737, Bug #30575163)

Changes in MySQL NDB Cluster 8.0.19 (2020-01-13, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change:** The default value for the `ndb_autoincrement_prefetch_sz` server system variable has been increased to 512. (Bug #30316314)
- **Important Change:** `NDB` now supports more than 2 fragment replicas (up to a maximum of 4). Setting `NoOfReplicas=3` or `NoOfReplicas=4` is now fully covered in our internal testing and thus supported for use in production. (Bug #97479, Bug #97579, Bug #25261716, Bug #30501414, Bug #30528105, WL #8426)
- **Important Change:** Added the `TransactionMemory` data node configuration parameter which simplifies configuration of data node memory allocation for transaction operations. This is part of ongoing work on pooling of transactional and Local Data Manager (LDM) memory.

  The following parameters are incompatible with `TransactionMemory` and cannot be set in the `config.ini` configuration file if this parameter has been set:

  - `MaxNoOfConcurrentIndexOperations`
  - `MaxNoOfFiredTriggers`
  - `MaxNoOfLocalOperations`
  - `MaxNoOfLocalScans`

  If you attempt to set any of these incompatible parameters concurrently with `TransactionMemory`, the cluster management server cannot start.

  For more information, see the description of the `TransactionMemory` parameter and Parameters incompatible with TransactionMemory. See also Data Node Memory Management, for information about how memory resources are allocated by NDB Cluster data nodes. (Bug #96995, Bug #30344471, WL #12687)
- **Important Change:** The maximum or default values for several NDB Cluster data node configuration parameters have been changed in this release. These changes are listed here:

  - The maximum value for `DataMemory` is increased from 1 terabyte to 16 TB.
  - The maximum value for `DiskPageBufferMemory` is also increased from 1 TB to 16 TB.
  - The default value for `StringMemory` is decreased to 5 percent. Previously, this was 25 percent.
  - The default value for `LcpScanProgressTimeout` is increased from 60 seconds to 180 seconds.

  (WL #13382)
- **Performance:** Read from any fragment replica, which greatly improves the performance of table reads at a very low cost to table write performance, is now enabled by default for all `NDB` tables. This means both that the default value for the `ndb_read_backup` system variable is now ON, and that the value of the `NDB_TABLE` comment option `READ_BACKUP` is 1 when creating a new `NDB` table. (Previously, the default values were OFF and 0, respectively.)

For more information, see Setting NDB Comment Options, as well as the description of the `ndb_read_backup` system variable. (WL #13383)

- **NDB Disk Data:** The latency of checkpoints for Disk Data files has been reduced when using non-volatile memory devices such as solid-state drives (especially those using NVMe for data transfer), separate physical drives for Disk Data files, or both. As part of this work, two new data node configuration parameters, listed here, have been introduced:

  - `MaxDiskDataLatency` sets a maximum on allowed latency for disk access, aborting transactions exceeding this amount of time to complete

  - `DiskDataUsingSameDisk` makes it possible to take advantage of keeping Disk Data files on separate disks by increasing the rate at which Disk Data checkpoints can be made

  This release also adds three new tables to the `ndbinfo` database. These tables, listed here, can assist with performance monitoring of Disk Data checkpointing:

  - `diskstat` provides information about Disk Data tablespace reads, writes, and page requests during the previous 1 second

  - `diskstats_1sec` provides information similar to that given by the `diskstat` table, but does so for each of the last 20 seconds

  - `pgman_time_track_stats` table reports on the latency of disk operations affecting Disk Data tablespaces

  For additional information, see Disk Data latency parameters. (WL #12924)

- Added the `ndb_metadata_sync` server system variable, which simplifies knowing when metadata synchronization has completed successfully. Setting this variable to `true` triggers immediate synchronization of all changes between the `NDB` dictionary and the MySQL data dictionary without regard to any values set for `ndb_metadata_check` or `ndb_metadata_check_interval`. When synchronization has completed, its value is automatically reset to `false`. (Bug #30406657)

- Added the `DedicatedNode` parameter for data nodes, API nodes, and management nodes. When set to true, this parameter prevents the management server from handing out this node's node ID to any node that does not request it specifically. Intended primarily for testing, this parameter may be useful in cases in which multiple management servers are running on the same host, and using the host name alone is not sufficient for distinguishing among processes of the same type. (Bug #91406, Bug #28239197)

- A stack trace is now written to the data node log on abnormal termination of a data node. (WL #13166)

- Automatic synchronization of metadata from the MySQL data dictionary to `NDB` now includes databases containing `NDB` tables. With this enhancement, if a table exists in `NDB`, and the table and the database it belongs to do not exist on a given SQL node, it is no longer necessary to create the database manually. Instead, the database, along with all `NDB` tables belonging to this database, should be created on the SQL node automatically. (WL #13490)

Bugs Fixed

- **Incompatible Change:** `ndb_restore` no longer restores shared users and grants to the `mysql.ndb_sql_metadata` table by default. A new command-line option `--include-stored-grants` is added to override this behavior and enable restoring of shared user and grant data and metadata.

  As part of this fix, `ndb_restore` can now also correctly handle an ordered index on a system table. (Bug #30237657)

References: See also: Bug #29534239, Bug #30459246.

- **Incompatible Change:** The minimum value for the `RedoOverCommitCounter` data node configuration parameter has been increased from 0 to 1. The minimum value for the `RedoOverCommitLimit` data node configuration parameter has also been increased from 0 to 1.

  You should check the cluster global configuration file and make any necessary adjustments to values set for these parameters before upgrading. (Bug #29752703)

- **macOS:** On macOS, SQL nodes sometimes shut down unexpectedly during the binary log setup phase when starting the cluster. This occurred when there existed schemas whose names used uppercase letters and `lower_case_table_names` was set to 2. This caused acquisition of metadata locks to be attempted using keys having the incorrect lettercase, and, subsequently, these locks to fail. (Bug #30192373)

- **Microsoft Windows; NDB Disk Data:** On Windows, restarting a data node other than the master when using Disk Data tables led to a failure in `TSMAN`. (Bug #97436, Bug #30484272)

- **Solaris:** When debugging, `ndbmtd` consumed all available swap space on Solaris 11.4 SRU 12 and later. (Bug #30446577)

- **Solaris:** The byte order used for numeric values stored in the `mysql.ndb_sql_metadata` table was incorrect on Solaris/Sparc. This could be seen when using `ndb_select_all` or `ndb_restore --print`. (Bug #30265016)

- **NDB Disk Data:** After dropping a disk data table on one SQL node, trying to execute a query against `INFORMATION_SCHEMA.FILES` on a different SQL node stalled at `Waiting for tablespace metadata lock`. (Bug #30152258)

  References: See also: Bug #29871406.

- **NDB Disk Data:** `ALTER TABLESPACE ... ADD DATAFILE` could sometimes hang while trying to acquire a metadata lock. (Bug #29871406)

- **NDB Disk Data:** Compatibility code for the Version 1 disk format used prior to the introduction of the Version 2 format in NDB 7.6 turned out not to be necessary, and is no longer used.

- Work done in NDB 8.0.18 to allow more nodes introduced long signal variants of several signals taking a bitmask as one of their arguments, and we started using these new long signal variants even if the previous (still supported) short variants would have been sufficient. This introduced several new opportunities for hitting `out of LongMessageBuffer` errors.

  To avoid this, now in such cases we use the short signal variants wherever possible. Some of the signals affected include `CM_REGCONF`, `CM_REGREF`, `FAIL_REP`, `NODE_FAILREP`, `ISOLATE_ORD`, `COPY_GCIREQ`, `START_RECREQ`, `NDB_STARTCONF`, and `START_LCP_REQ`. (Bug #30708009)

  References: See also: Bug #30707970.

- The fix made in NDB 8.0.18 for an issue in which a transaction was committed prematurely aborted the transaction if the table definition had changed midway, but failed in testing to free memory allocated by `getExtraMetadata()`. Now this memory is properly freed before aborting the transaction. (Bug #30576983)

  References: This issue is a regression of: Bug #29911440.

- Excessive allocation of attribute buffer when initializing data in `DBTC` led to preallocation of api connection records failing due to unexpectedly running out of memory. (Bug #30570264)

- Improved error handling in the case where `NDB` attempted to update a local user having the `NDB_STORED_USER` privilege but which could not be found in the `ndb_sql_metadata` table. (Bug #30556487)

- Failure of a transaction during execution of an `ALTER TABLE ... ALGORITHM=COPY` statement following the rename of the new table to the name of the original table but before dropping the original table caused `mysqld` to exit prematurely. (Bug #30548209)
- Non-MSI builds on Windows using `-DWITH_NDBCLUSTER` did not succeed unless the WiX toolkit was installed. (Bug #30536837)
- The `allowed_values` output from `ndb_config --xml --configinfo` for the `Arbitration` data node configuration parameter in NDB 8.0.18 was not consistent with that obtained in previous releases. (Bug #30529220)

  References: See also: Bug #30505003.
- A faulty `ndbrequire()` introduced when implementing partial local checkpoints assumed that `m_participatingLQH` must be clear when receiving `START_LCP_REQ`, which is not necessarily true when a failure happens for the master after sending `START_LCP_REQ` and before handling any `START_LCP_CONF` signals. (Bug #30523457)
- A local checkpoint sometimes hung when the master node failed while sending an `LCP_COMPLETE_REP` signal and it was sent to some nodes, but not all of them. (Bug #30520818)
- Added the `DUMP 9988` and `DUMP 9989` commands. (Bug #30520103)
- The management server did not handle all cases of `NODE_FAILREP` correctly. (Bug #30520066)
- With `SharedGlobalMemory` set to 0, some resources did not meet required minimums. (Bug #30411835)
- Execution of `ndb_restore --rebuild-indexes` together with the `--rewrite-database` and `--exclude-missing-tables` options did not create indexes for any tables in the target database. (Bug #30411122)
- When writing the schema operation into the `ndb_schema` table failed, the states in the `NDB_SCHEMA` object were not cleared, which led to the SQL node shutting down when it tried to free the object. (Bug #30402362)

  References: See also: Bug #30371590.
- When synchronizing extent pages it was possible for the current local checkpoint (LCP) to stall indefinitely if a `CONTINUEB` signal for handling the LCP was still outstanding when receiving the `FSWRITECONF` signal for the last page written in the extent synchronization page. The LCP could also be restarted if another page was written from the data pages. It was also possible that this issue caused `PREP_LCP` pages to be written at times when they should not have been. (Bug #30397083)
- If a transaction was aborted while getting a page from the disk page buffer and the disk system was overloaded, the transaction hung indefinitely. This could also cause restarts to hang and node failure handling to fail. (Bug #30397083, Bug #30360681)

  References: See also: Bug #30152258.
- Data node failures with the error `Another node failed during system restart...` occurred during a partial restart. (Bug #30368622)
- Automatic synchronization could potentially trigger an increase in the number of locks being taken on a particular metadata object at a given time, such as when a synchronization attempt coincided with a DDL or DML statement involving the same metadata object; competing locks could lead to the NDB deadlock detection logic penalizing the user action rather than the background synchronization. We fix this by changing all exclusive metadata lock acquisition attempts during auto-synchronization so that they use a timeout of 0 (rather than the 10 seconds previously allowed), which avoids deadlock detection and gives priority to the user action. (Bug #30358470)
- If a `SYNC_EXTENT_PAGES_REQ` signal was received by `PGMAN` while dropping a log file group as part of a partial local checkpoint, and thus dropping the page locked by this block for processing

next, the LCP terminated due to trying to access the page after it had already been dropped. (Bug #30305315)

- The wrong number of bytes was reported in the cluster log for a completed local checkpoint. (Bug #30274618)

  References: See also: Bug #29942998.

- Added the new `ndb_mgm` client debugging commands `DUMP 2356` and `DUMP 2357`. (Bug #30265415)

- Executing `ndb_drop_table` using the `--help` option caused this program to terminate prematurely, and without producing any help output. (Bug #30259264)

- A `mysqld` trying to connect to the cluster, and thus trying to acquire the global schema lock (GSL) during setup, ignored the setting for `ndb-wait-setup` and hung indefinitely when the GSL had already been acquired by another `mysqld`, such as when it was executing an `ALTER TABLE` statement. (Bug #30242141)

- When a table containing self-referential foreign key (in other words, a foreign key referencing another column of the same table) was altered using the `COPY` algorithm, the foreign key definition was removed. (Bug #30233405)

- In MySQL 8.0, names of foreign keys explicitly provided by user are generated automatically in the SQL layer and stored in the data dictionary. Such names are of the form `[table_name]_ibfk_[#]` which align with the names generated by the `InnoDB` storage engine in MySQL 5.7. NDB 8.0.18 introduced a change in behavior by `NDB` such that it also uses the generated names, but in some cases, such as when tables were renamed, `NDB` still generated and used its own format for such names internally rather than those generated by the SQL layer and stored in the data dictionary, which led to the following issues:

  - Discrepancies in `SHOW CREATE TABLE` output and the contents of `INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS`

  - Improper metadata locking for foreign keys

  - Confusing names for foreign keys in error messages

  Now `NDB` also renames the foreign keys in such cases, using the names provided by the MySQL server, to align fully with those used by `InnoDB`. (Bug #30210839)

  References: See also: Bug #96508, Bug #30171959.

- When a table referenced by a foreign key was renamed, participating SQL nodes did not properly update the foreign key definitions for the referencing table in their data dictionaries during schema distribution. (Bug #30191068)

- Data node handling of failures of other data nodes could sometimes not be synchronized properly, such that two or more data nodes could see different nodes as the master node. (Bug #30188414)

- Some scan operations failed due to the presence of an old assert in `DbtupBuffer.cpp` that checked whether API nodes were using a version of the software previous to NDB 6.4. This was no longer necessary or correct, and has been removed. (Bug #30188411)

- When executing a global schema lock (GSL), `NDB` used a single `Ndb_table_guard` object for successive retires when attempting to obtain a table object reference; it was not possible for this to succeed after failing on the first attempt, since `Ndb_table_guard` assumes that the underlying

object pointer is determined once only—at initialisation—with the previously retrieved pointer being returned from a cached reference thereafter.

This resulted in infinite waits to obtain the GSL, causing the binlog injector thread to hang so that `mysqld` considered all `NDB` tables to be read-only. To avoid this problem, `NDB` now uses a fresh instance of `Ndb_table_guard` for each such retry. (Bug #30120858)

References: This issue is a regression of: Bug #30086352.

- When upgrading an SQL node to NDB 8.0 from a previous release series, the `.frm` file whose contents are read and then installed in the data dictionary does not contain any information about foreign keys. This meant that foreign key information was not installed in the SQL node's data dictionary. This is fixed by using the foreign key information available in the NDB data dictionary to update the local MySQL data dictionary during table metadata upgrade. (Bug #30071043)

- Restoring tables with the `--disable-indexes` option resulted in the wrong table definition being installed in the MySQL data dictionary. This is because the serialized dictionary information (SDI) packed into the NDB dictionary's table definition is used to create the table object; the SDI definition is updated only when the DDL change is done through the MySQL server. Installation of the wrong table definition meant that the table could not be opened until the indexes were re-created in the NDB dictionary again using `--rebuild-indexes`.

  This is fixed by extending auto-synchronization such that it compares the SDI to the NDB dictionary table information and fails in cases in which the column definitions do not match. Mismatches involving indexes only are treated as temporary errors, with the table in question being detected again during the next round of change detection. (Bug #30000202, Bug #30414514)

- Restoring tables for which `MAX_ROWS` was used to alter partitioning from a backup made from NDB 7.4 to a cluster running NDB 7.6 did not work correctly. This is fixed by ensuring that the upgrade code handling `PartitionBalance` supplies a valid table specification to the `NDB` dictionary. (Bug #29955656)

- The number of data bytes for the summary event written in the cluster log when a backup completed was truncated to 32 bits, so that there was a significant mismatch between the number of log records and the number of data records printed in the log for this event. (Bug #29942998)

- `mysqld` sometimes aborted during a long `ALTER TABLE` operation that timed out. (Bug #29894768)

  References: See also: Bug #29192097.

- When an SQL node connected to `NDB`, it did not know whether it had previously connected to that cluster, and thus could not determine whether its data dictionary information was merely out of date, or completely invalid. This issue is solved by implementing a unique schema version identifier (schema UUID) to the `ndb_schema` table in `NDB` as well as to the `ndb_schema` table object in the data dictionary. Now, whenever a `mysqld` connects to a cluster as an SQL node, it can compare the schema UUID stored in its data dictionary against that which is stored in the `ndb_schema` table, and so know whether it is connecting for the first time. If so, the SQL node removes any entries that may be in its data dictionary. (Bug #29894166)

  References: See also: Bug #27543602.

- Improved log messages generated by table discovery and table metadata upgrades. (Bug #29894127)

- Using 2 LDM threads on a 2-node cluster with 10 threads per node could result in a partition imbalance, such that one of the LDM threads on each node was the primary for zero fragments. Trying to restore a multi-threaded backup from this cluster failed because the datafile for one LDM contained only the 12-byte data file header, which `ndb_restore` was unable to read. The same

problem could occur in other cases, such as when taking a backup immediately after adding an empty node online.

It was found that this occurred when `ODirect` was enabled for an EOF backup data file write whose size was less than 512 bytes and the backup was in the `STOPPING` state. This normally occurs only for an aborted backup, but could also happen for a successful backup for which an LDM had no fragments. We fix the issue by introducing an additional check to ensure that writes are skipped only if the backup actually contains an error which should cause it to abort. (Bug #29892660)

References: See also: Bug #30371389.

- For `NDB` tables, `ALTER TABLE ... ALTER INDEX` did not work with `ALGORITHM=INPLACE`. (Bug #29700197)
- `ndb_restore` failed in testing on 32-bit platforms. This issue is fixed by increasing the size of the thread stack used by this tool from 64 KB to 128 KB. (Bug #29699887)

  References: See also: Bug #30406046.
- An unplanned shutdown of the cluster occurred due to an error in `DBTUP` while deleting rows from a table following an online upgrade. (Bug #29616383)
- In some cases the `SignalSender` class, used as part of the implementation of `ndb_mgmd` and `ndbinfo`, buffered excessive numbers of unneeded `SUB_GCP_COMPLETE_REP` and `API_REGCONF` signals, leading to unnecessary consumption of memory. (Bug #29520353)

  References: See also: Bug #20075747, Bug #29474136.
- The setting for the `BackupLogBufferSize` configuration parameter was not honored. (Bug #29415012)
- When `mysqld` was run with the `--upgrade=FORCE` option, it reported the following issues:

  ```
  [Warning]  Table 'mysql.ndb_apply_status' requires repair.
  [ERROR]    Table 'mysql.ndb_apply_status' repair failed.
  ```

  This was because `--upgrade=FORCE` causes a bootstrap system thread to run `CHECK TABLE FOR UPGRADE`, but `ha_ndbcluster::open()` refused to open the table before schema synchronization had completed, which eventually led to the reported conditions. (Bug #29305977)

  References: See also: Bug #29205142.
- When using explicit SHM connections, with `ShmSize` set to a value larger than the system's available shared memory, `mysqld` hung indefinitely on startup and produced no useful error messages. (Bug #28875553)
- The maximum global checkpoint (GCP) commit lag and GCP save timeout are recalculated whenever a node shuts down, to take into account the change in number of data nodes. This could lead to the unintentional shutdown of a viable node when the threshold decreased below the previous value. (Bug #27664092)

  References: See also: Bug #26364729.
- A transaction which inserts a child row may run concurrently with a transaction which deletes the parent row for that child. One of the transactions should be aborted in this case, lest an orphaned child row result.

  Before committing an insert on a child row, a read of the parent row is triggered to confirm that the parent exists. Similarly, before committing a delete on a parent row, a read or scan is performed to confirm that no child rows exist. When insert and delete transactions were run concurrently, their prepare and commit operations could interact in such a way that both transactions committed. This occurred because the triggered reads were performed using `LM_CommittedRead` locks (see `NdbOperation::LockMode`), which are not strong enough to prevent such error scenarios.

This problem is fixed by using the stronger `LM_SimpleRead` lock mode for both triggered reads. The use of `LM_SimpleRead` rather than `LM_CommittedRead` locks ensures that at least one transaction aborts in every possible scenario involving transactions which concurrently insert into child rows and delete from parent rows. (Bug #22180583)

- Concurrent `SELECT` and `ALTER TABLE` statements on the same SQL node could sometimes block one another while waiting for locks to be released. (Bug #17812505, Bug #30383887)
- Failure handling in schema synchronization involves pushing warnings and errors to the binary logging thread. Schema synchronization is also retried in case of certain failures which could lead to an accumulation of warnings in the thread. Now such warnings and errors are cleared following each attempt at schema synchronization. (Bug #2991036)
- An `INCL_NODECONF` signal from any local blocks should be ignored when a node has failed, except in order to reset `c_nodeStartSlave.nodeId`. (Bug #96550, Bug #30187779)
- When returning Error 1022, `NDB` did not print the name of the affected table. (Bug #74218, Bug #19763093)

  References: See also: Bug #29700174.

Changes in MySQL NDB Cluster 8.0.18 (2019-10-14, Release Candidate)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change:** The 63-byte limit on `NDB` database and table names has been removed. These identifiers may now take up to 64 bytes, as when using other MySQL storage engines. For more information, see Previous NDB Cluster Issues Resolved in NDB Cluster 8.0. (Bug #44940, Bug #11753491, Bug #27447958)
- **Important Change:** Implemented the `NDB_STORED_USER` privilege, which enables sharing of users, roles, and privileges across all SQL nodes attached to a given NDB Cluster. This replaces the distributed grant tables mechanism from NDB 7.6 and earlier versions of NDB Cluster, which was removed in NDB 8.0.16 due to its incompatibility with changes made to the MySQL privilege system in MySQL 8.0.

  A user or role which has this privilege is propagated, along with its (other) privileges to a MySQL server (SQL node) as soon as it connects to the cluster. Changes made to the privileges of the user or role are synchronized immediately with all connected SQL nodes.

  `NDB_STORED_USER` can be granted to users and roles other than reserved accounts such as `mysql.session@localhost` or `mysql.infoschema@localhost`. A role can be shared, but assigning a shared role to a user does not cause this user to be shared; the `NDB_STORED_USER` privilege must be granted to the user explicitly in order for the user to be shared between NDB Cluster SQL nodes.

  The `NDB_STORED_USER` privilege is always global and must be granted using `ON *.*`. This privilege is recognized only if the MySQL server enables support for the `NDBCLUSTER` storage engine.

  For usage information, see the description of `NDB_STORED_USER`. Privilege Synchronization and NDB_STORED_USER, has additional information on how `NDB_STORED_USER` and privilege synchronization work. For information on how this change may affect upgrades to NDB 8.0 from previous versions, see Upgrading and Downgrading NDB Cluster. (WL #12637)

  References: See also: Bug #29862601, Bug #29996547.

- **Important Change:** The maximum row size for an `NDB` table is increased from 14000 to 30000 bytes.

  As before, only the first 264 bytes of a `BLOB` or `TEXT` column count towards this total.

  The maximum offset for a fixed-width column of an `NDB` table is 8188 bytes; this is also unchanged from previous NDB Cluster releases.

  For more information, see Limits Associated with Database Objects in NDB Cluster. (WL #13079, WL #11160)

  References: See also: Bug #29485977, Bug #29024275.

- **Important Change:** A new binary format has been implemented for the NDB management server's cached configuration file, which is intended to support much larger numbers of nodes in a cluster than previously. Prior to this release, the configuration file supported a maximum of 16381 sections; this number is increased to 4G.

  Upgrades to the new format should not require any manual intervention, as the management server (and other cluster nodes) can still read the old format. For downgrades from this release or a later one to NDB 8.0.17 or earlier, it is necessary to remove the binary configuration files prior to starting the old management server binary, or start it using the `--initial` option.

  For more information, see Upgrading and Downgrading NDB Cluster. (WL #12453)

- **Important Change:** The maximum number of data nodes supported in a single NDB cluster is raised in this release from 48 to 144. The range of supported data node IDs is increased in conjunction with this enhancement to 1-144, inclusive.

  In previous releases, recommended node IDs for management nodes were 49 and 50. These values are still supported, but, if used, limit the maximum number of data nodes to 142. For this reason, the recommended node ID values for management servers are now 145 and 146.

  The maximum total supported number of nodes of all types in a given cluster is 255. This total is unchanged from previous releases.

  For a cluster running more than 48 data nodes, it is not possible to downgrade directly to a previous release that supports only 48 data nodes. In such cases, it is necessary to reduce the number of data nodes to 48 or fewer, and to make sure that all data nodes use node IDs that are less than 49.

  This change also introduces a new version (v2) of the format used for the data node `sysfile`, which records information such as the last global checkpoint index, restart status, and node group membership of each node (see NDB Cluster Data Node File System Directory). (WL #12680, WL #12564, WL #12876)

- **NDB Cluster APIs:** An alternative constructor for `NdbInterpretedCode` is now provided, which accepts an `NdbRecord` in place of a `Table` object. (Bug #29852377)

- **NDB Cluster APIs:** `NdbScanFilter::cmp()` and the following `NdbInterpretedCode` comparison methods can be now used to compare table column values:

  - `branch_col_eq()`
  - `branch_col_ge()`
  - `branch_col_gt()`
  - `branch_col_le()`
  - `branch_col_lt()`
  - `branch_col_ne()`

When using any of these methods, the table column values to be compared must be of exactly the same type, including with respect to length, precision, and scale. In addition, in all cases, NULL is always considered by these methods to be less than any other value. You should also be aware that, when used to compare table column values, NdbScanFilter::cmp() does not support all possible values of BinaryCondition.

For more information, see the descriptions of the individual API methods. (WL #13120)

- **NDB Client Programs:** The dependency of the ndb_delete_all utility on the NDBT library has been removed. This library, used in NDB development for testing, is not required for normal use. The visible change for users is that ndb_delete_all no longer prints NDBT_ProgramExit - *status* following completion of its run. Applications that depend upon this behavior should be updated to reflect this change when upgrading to this release. (WL #13223)

- ndb_restore now reports the specific NDB error number and message when it is unable to load a table descriptor from a backup .ctl file. This can happen when attempting to restore a backup taken from a later version of the NDB Cluster software to a cluster running an earlier version—for example, when the backup includes a table using a character set which is unknown to the version of ndb_restore being used to restore it. (Bug #30184265)

- The output from DUMP 1000 in the ndb_mgm client has been extended to provide information regarding total data page usage. (Bug #29841454)

  References: See also: Bug #29929996.

- NDB Cluster's condition pushdown functionality has been extended as follows:

  - Expressions using any previously allowed comparisons are now supported.

  - Comparisons between columns in the same table and of the same type are now supported. The columns must be of exactly the same type.

  Example: Suppose there are two tables t1 and t2 created as shown here:

```
CREATE TABLE t1 (a INT, b INT, c CHAR(10), d CHAR(5)) ENGINE=NDB;
CREATE TABLE t2 LIKE t1;
```

  The following joins can now be pushed down to the data nodes:

```
SELECT * FROM t1 JOIN t2 ON t2.a < t1.a+10;
SELECT * FROM t1 JOIN t2 ON t2.a = t1.a+t1.b;
SELECT * FROM t1 JOIN t2 ON t2.a = t1.a+t1.b;
SELECT * FROM t1 JOIN t2 ON t2.d = SUBSTRING(t1.c,1,5);
SELECT * FROM t1 JOIN t2 ON t2.c = CONCAT('foo',t1.d,'ba');
```

  Supported comparisons are <, <=, >, >=, =, and <>. (Bug #29685643, WL #12956, WL #13121)

- NDB Cluster now uses *table_name*_fk_*N* as the naming pattern for internally generated foreign keys, which is similar to the *table_name*_ibfk_*N* pattern used by InnoDB. (Bug #96508, Bug #30171959)

  References: See also: Bug #30210839.

- Added the ndb_schema_dist_lock_wait_timeout system variable to control how long to wait for a schema lock to be released when trying to update the SQL node's local data dictionary for one or more tables currently in use from the NDB data dictionary's metadata. If this synchronization has not yet occurred by the end of this time, the SQL node returns a warning that schema distribution did not succeed; the next time that the table for which distribution failed is accessed, NDB tries once again to synchronize the table metadata. (WL #10164)

- NDB table objects submitted by the metadata change monitor thread are now automatically checked for any mismatches and synchronized by the NDB binary logging thread. The status variable

`Ndb_metadata_synced_count` added in this release shows the number of objects synchronized automatically; it is possible to see which objects have been synchronized by checking the cluster log. In addition, the new status variable `Ndb_metadata_blacklist_size` indicates the number of objects for which synchronization has failed. (WL #11914)

References: See also: Bug #30000202.

- It is now possible to build `NDB` for 64-bit `ARM` CPUs from the NDB Cluster sources. Currently, we do not provide any precompiled binaries for this platform. (WL #12928)
- Start times for the `ndb_mgmd` management node daemon have been significantly improved as follows:
  - More efficient handling of properties from configuration data can decrease startup times for the management server by a factor of 6 or more as compared with previous versions.
  - Host names not present in the management server's `hosts` file no longer create a bottleneck during startup, making `ndb_mgmd` start times up to 20 times shorter where these are used.

  (WL #13143)
- Columns of `NDB` tables can now be renamed online, using `ALGORITHM=INPLACE`. (WL #11734)

  References: See also: Bug #28609968.

Bugs Fixed

- **Important Change:** Because the current implementation for node failure handling cannot guarantee that even a single transaction of size `MaxNoOfConcurrentOperations` is completed in each round, this parameter is once again used to set a global limit on the total number of concurrent operations in all transactions within a single transaction coordinator instance. (Bug #96617, Bug #30216204)
- **Partitioning; NDB Disk Data:** Creation of a partitioned disk data table was unsuccessful due to a missing metadata lock on the tablespace specified in the `CREATE TABLE` statement. (Bug #28876892)
- **NDB Disk Data:** Tablespaces and data files are not tightly coupled in `NDB`, in the sense that they are represented by independent `NdbDictionary` objects. Thus, when metadata is restored using the `ndb_restore` tool, there was no guarantee that the tablespace and its associated datafile objects were restored at the same time. This led to the possibility that the tablespace mismatch was detected and automatically synchronized to the data dictionary before the datafile was restored to `NDB`. This issue also applied to log file groups and undo files.

  To fix this problem, the metadata change monitor now submits tablespaces and logfile groups only if their corresponding datafiles and undofiles actually exist in `NDB`. (Bug #30090080)
- **NDB Disk Data:** When a data node failed following creation and population of an `NDB` table having columns on disk, but prior to execution of a local checkpoint, it was possible to lose row data from the tablespace. (Bug #29506869)
- **NDB Cluster APIs:** The NDB API examples `ndbapi_array_simple.cpp` (see NDB API Simple Array Example) and `ndbapi_array_using_adapter.cpp` (see NDB API Simple Array Example Using Adapter) made assignments directly to a `std::vector` array instead of using `push_back()` calls to do so. (Bug #28956047)
- **MySQL NDB ClusterJ:** If ClusterJ was deployed as a separate module of a multi-module web application, when the application tried to create a new instance of a domain object, the exception `java.lang.IllegalArgumentException: non-public interface is not defined by the given loader` was thrown. It was because ClusterJ always tries to create a proxy class from which the domain object can be instantiated, and the proxy class is an implementation of the domain interface and the protected `DomainTypeHandlerImpl::Finalizable` interface.

The class loaders of these two interfaces were different in the case, as they belonged to different modules running on the web server, so that when ClusterJ tried to create the proxy class using the domain object interface's class loader, the above-mentioned exception was thrown. This fix makes the `Finalization` interface public so that the class loader of the web application would be able to access it even if it belongs to a different module from that of the domain interface. (Bug #29895213)

- **MySQL NDB ClusterJ:** ClusterJ sometimes failed with a segmentation fault after reconnecting to an NDB Cluster. This was due to ClusterJ reusing old database metadata objects from the old connection. With the fix, those objects are discarded before a reconnection to the cluster. (Bug #29891983)
- Faulty calculation of microseconds caused the internal `ndb_milli_sleep()` function to sleep for too short a time. (Bug #30211922)
- Once a data node is started, 95% of its configured `DataMemory` should be available for normal data, with 5% to spare for use in critical situations. During the node startup process, all of its configured `DataMemory` is usable for data, in order to minimize the risk that restoring the node data fails due to running out of data memory due to some dynamic memory structure using more pages for the same data than when the node was stopped. For example, a hash table grows differently during a restart than it did previously, since the order of inserts to the table differs from the historical order.

  The issue raised in this bug report occurred when a check that the data memory used plus the spare data memory did not exceed the value set for `DataMemory` failed at the point where the spare memory was reserved. This happened as the state of the data node transitioned from starting to started, when reserving spare pages. After calculating the number of reserved pages to be used for spare memory, and then the number of shared pages (that is, pages from shared global memory) to be used for this, the number of reserved pages already allocated was not taken into consideration. (Bug #30205182)

  References: See also: Bug #29616383.
- Removed a memory leak found in the `ndb_import` utility. (Bug #30192989)
- It was not possible to use `ndb_restore` and a backup taken from an NDB 8.0 cluster to restore to a cluster running NDB 7.6. (Bug #30184658)

  References: See also: Bug #30221717.
- When starting, a data node's local sysfile was not updated between the first completed local checkpoint and start phase 50. (Bug #30086352)
- In the `BACKUP` block, the assumption was made that the first record in `c_backups` was the local checkpoint record, which is not always the case. Now `NDB` loops through the records in `c_backups` to find the (correct) LCP record instead. (Bug #30080194)
- During node takeover for the master it was possible to end in the state `LCP_STATUS_IDLE` while the remaining data nodes were reporting their state as `LCP_TAB_SAVED`. This led to failure of the node when attempting to handle reception of a `LCP_COMPLETE_REP` signal since this is not expected when idle. Now in such cases local checkpoint handling is done in a manner that ensures that this node finishes in the proper state (`LCP_TAB_SAVED`). (Bug #30032863)
- When a MySQL Server built with `NDBCLUSTER` support was run on Solaris/x86, it failed during schema distribution. The root cause of the problem was an issue with the Developer Studio compiler used to build binaries for this platform when optimization level `-xO2` was used. This issue is fixed by using optimization level `-xO1` instead for `NDBCLUSTER` built for Solaris/x86. (Bug #30031130)

  References: See also: Bug #28585914, Bug #30014295.
- `NDB` used `free()` directly to deallocate `ndb_mgm_configuration` objects instead of calling `ndb_mgm_destroy_configuration()`, which correctly uses `delete` for deallocation. (Bug #29998980)

- Default configuration sections did not have the configuration section types set when unpacked into memory, which caused a memory leak since this meant that the section destructor would not destroy the entries for these sections. (Bug #29965125)
- No error was propagated when `NDB` failed to discover a table due to the table format being old and no longer supported, which could cause the `NDB` handler to retry the discovery operation endlessly and thereby hang. (Bug #29949096, Bug #29934763)
- During upgrade of an NDB Cluster when half of the data nodes were running NDB 7.6 while the remainder were running NDB 8.0, attempting to shut down those nodes which were running NDB 7.6 led to failure of one node with the error `CHECK FAILEDNODEPTR.P->DBLQHFAI`. (Bug #29912988, Bug #30141203)
- Altering a table in the middle of an ongoing transaction caused a table discovery operation which led to the transaction being committed prematurely; in addition, no error was returned when performing further updates as part of the same transaction.

  Now in such cases, the table discovery operation fails, when a transaction is in progress. (Bug #29911440)
- When performing a local checkpoint (LCP), a table's schema version was intermittently read as 0, which caused `NDB` LCP handling to treat the table as though it were being dropped. This could effect rebuilding of indexes offline by `ndb_restore` while the table was in the `TABLE_READ_ONLY` state. Now the function reading the schema version (`getCreateSchemaVersion()`) no longer not changes it while the table is read-only. (Bug #29910397)
- When an error occurs on an SQL node during schema distribution, information about this was written in the error log, but no indication was provided by the `mysql` client that the DDL statement in question was unsuccessful. Now in such cases, one or more generic warnings are displayed by the client to indicate that a given schema distribution operation has not been successful, with further information available in the error log of the originating SQL node. (Bug #29889869)
- Errors and warnings pushed to the execution thread during metadata synchronization and metadata change detection were not properly logged and cleared. (Bug #29874313)
- Altering a normal column to a stored generated column was performed online even though this is not supported. (Bug #29862463)
- A pushed join with `ORDER BY` did not always return the rows of the result in the specified order. This could occur when the optimizer used an ordered index to provide the ordering and the index used a column from the table that served as the root of the pushed join. (Bug #29860378)
- A number of issues in the Backup block for local checkpoints (LCPs) were found and fixed, including the following:
  - Bytes written to LCP part files were not always included in the LCP byte count.
  - The maximum record size for the buffer used for all LCP part files was not updated in all cases in which the table maximum record size had changed.
  - LCP surfacing could occur for LCP scans at times other than when receiving `SCAN_FRAGCONF` signals.
  - It was possible in some cases for the table currently being scanned to be altered in the middle of a scan request, which behavior is not supported.

  (Bug #29843373)

  References: See also: Bug #29485977.
- The `requestInfo` fields for the long and short forms of the `LQHKEYREQ` signal had different definitions; bits used for the key length in the short version were reused for flags in the long version,

since the key length is implicit in the section length of the long version of the signal but it was possible for long LQHKEYREQ signals to contain a keylength in these same bits, which could be misinterpreted by the receiving local query handler, potentially leading to errors. Checks have now been implemented to make sure that this no longer happens. (Bug #29820838)

- The list of dropped shares could hold only one dropped NDB_SHARE instance for each key, which prevented NDB_SHARE instances with same key from being dropped multiple times while handlers held references to those NDB_SHARE instances. This interfered with keeping track of the memory allocated and being able to release it if mysqld shut down without all handlers having released their references to the shares. To resolve this issue, the dropped share list has been changed to use a list type which allows more than one NDB_SHARE with the same key to exist at the same time. (Bug #29812659, Bug #29812613)

- Removed an ndb_restore compile-time dependency on table names that was defined by the ndbcluster plugin. (Bug #29801100)

- When creating a table in parallel on multiple SQL nodes, the result was a race condition between checking that the table existed and opening the table, which caused CREATE TABLE IF NOT EXISTS to fail with Error 1. This was the result of two issues, described with their fixes here:

  1. Opening a table whose NDB_SHARE did not exist returned the non-descriptive error message ERROR 1296 (HY000): Got error 1 'Unknown error code' from NDBCLUSTER. This is fixed with a warning describing the problem in more detail, along with a more sensible error code.

     It was possible to open a table before schema synchronization was completed. This is fixed with a warning better describing the problem, along with an error indicating that cluster is not yet ready.

  In addition, this fixes a related issue in which creating indexes sometimes also failed with Error 1. (Bug #29793534, Bug #29871321)

- Previously, for a pushed condition, every request sent to NDB for a given table caused the generation of a new instance of NdbInterpretedCode. When joining tables, generation of multiple requests for all tables following the first table in the query plan is very likely; if the pushed condition had no dependencies on prior tables in the query plan, identical instances of NdbInterpretedCode were generated for each request, at a significant cost in wasted CPU cycles. Now such pushed conditions are identified and the required NdbInterpretedCode object is generated only once, and reused for every request sent for this table without the need for generating new code each time.

  This change also makes it possible for Scan Filter too large errors to be detected and set during query optimization, which corrects cases where the query plan shown was inaccurate because the indicated push of a condition later had to be undone during the execution phase. (Bug #29704575)

- Some instances of NdbScanFilter used in pushdown conditions were not generated properly due to FLOAT values being represented internally as having zero length. This led to more than the expected number of rows being returned from NDB, as shown by the value of Ndb_api_read_row_count. While the condition was re-evaluated by mysqld when generation of scan filter failed, the end result was still correct in such cases, but any performance gain expected from pushing the condition was lost. (Bug #29699347)

- When creating a table, NDB did not always determine correctly whether it exceeded the maximum allowed record size. (Bug #29698277)

- NDB index statistics are calculated based on the topology of one fragment of an ordered index; the fragment chosen in any particular index is decided at index creation time, both when the index is originally created, and when a node or system restart has recreated the index locally. This calculation is based in part on the number of fragments in the index, which can change when a table is reorganized. This means that, the next time that the node is restarted, this node may choose a

different fragment, so that no fragments, one fragment, or two fragments are used to generate index statistics, resulting in errors from `ANALYZE TABLE`.

This issue is solved by modifying the online table reorganization to recalculate the chosen fragment immediately, so that all nodes are aligned before and after any subsequent restart. (Bug #29534647)

- As part of initializing schema distribution, each data node must maintain a subscriber bitmap providing information about the API nodes that are currently subscribed to this data node. Previously, the size of the bitmap was hard-coded to `MAX_NODES` (256), which meant that large amounts of memory might be allocated but never used when the cluster had significantly fewer nodes than this value. Now the size of the bitmap is determined by checking the maximum API node ID used in the cluster configuration file. (Bug #29270539)

- The removal of the `mysql_upgrade` utility and its replacement by `mysqld --initialize` means that the upgrade procedure is executed much earlier than previously, possibly before `NDB` is fully ready to handle queries. This caused migration of the MySQL privilege tables from `NDB` to `InnoDB` to fail. (Bug #29205142)

- During a restart when the data nodes had started but not yet elected a president, the management server received a `node ID already in use` error, which resulted in excessive retries and logging. This is fixed by introducing a new error 1705 `Not ready for connection allocation yet` for this case.

  During a restart when the data nodes had not yet completed node failure handling, a spurious `Failed to allocate nodeID` error was returned. This is fixed by adding a check to detect an incomplete node start and to return error 1703 `Node failure handling not completed` instead.

  As part of this fix, the frequency of retries has been reduced for `not ready to alloc nodeID` errors, an error insert has been added to simulate a slow restart for testing purposes, and log messages have been reworded to indicate that the relevant node ID allocation errors are minor and only temporary. (Bug #27484514)

- `NDB` on Windows and macOS platforms did not always treat table names using mixed case consistently with `lower_case_table_names` = 2. (Bug #27307793)

- The process of selecting the transaction coordinator checked for “live” data nodes but not necessarily for those that were actually available. (Bug #27160203)

- The automatic metadata synchronization mechanism requires the binary logging thread to acquire the global schema lock before an object can be safely synchronized. When another thread had acquired this lock at the same time, the binary logging thread waited for up to `TransactionDeadlockDetectionTimeout` milliseconds and then returned failure if it was unsuccessful in acquiring the lock, which was unnecessary and which negatively impacted performance.

  This has been fixed by ensuring that the binary logging thread acquires the global schema lock, or else returns with an error, immediately. As part of this work, a new `OperationOptions` flag `OO_NOWAIT` has also been implemented in the NDB API. (WL #29740946)

Changes in MySQL NDB Cluster 8.0.17 (2019-07-22, Release Candidate)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Schema operation timeout detection has been moved from the schema distribution client to the schema distribution coordinator, which now checks ongoing schema operations for timeout at regular

intervals, marks participants that have timed out, emits suitable warnings when a schema operation timeout occurs, and prints a list of any ongoing schema operations at regular intervals.

As part of this work, a new option `--ndb-schema-dist-timeout` makes it possible to set the number of seconds for a given SQL node to wait until a schema operation is marked as having timed out. (Bug #29556148)

- Added the status variable `Ndb_trans_hint_count_session`, which shows the number of transactions started in the current session that used hints. Compare this with `Ndb_api_trans_start_count_session` to get the proportion of all `NDB` transactions in the current session that have been able to use hinting. (Bug #29127040)
- When the cluster is in single user mode, the output of the `ndb_mgm SHOW` command now indicates which API or SQL node has exclusive access while this mode is in effect. (Bug #16275500)

Bugs Fixed

- **Important Change:** Attempting to drop, using the `mysql` client, an `NDB` table that existed in the MySQL data dictionary but not in `NDB` caused `mysqld` to fail with an error. This situation could occur when an `NDB` table was dropped using the `ndb_drop_table` tool or in an NDB API application using `dropTable()`. Now in such cases, `mysqld` drops the table from the MySQL data dictionary without raising an error. (Bug #29125206)
- **Important Change:** The dependency of `ndb_restore` on the `NDBT` library, which is used for internal testing only, has been removed. This means that the program no longer prints `NDBT_ProgramExit: ...` when terminating. Applications that depend upon this behavior should be updated to reflect this change when upgrading to this release. (WL #13117)
- **Packaging:** Added debug symbol packages to `NDB` distributions for `.deb`-based platforms which do not generate these automatically. (Bug #29040024)
- **NDB Disk Data:** If, for some reason, a disk data table exists in the NDB data dictionary but not in that of the MySQL server, the data dictionary is synchronized by installing the object. This can occur either during the schema synchronization phase when a MySQL server connects to an NDB Cluster, or during table discovery through a DML query or DDL statement.

  For disk data tables which used a tablespace for storage, the tablespace ID is stored as part of the data dictionary object, but this was not set during synchronization. (Bug #29597249)
- **NDB Disk Data:** Concurrent Disk Data table and tablespace DDL statements executed on the same SQL node caused a metadata lock deadlock. A DDL statement requires that an exclusive lock be taken on the object being modified and every such lock in turn requires that the global schema lock be acquired in `NDB`.

  To fix this issue, `NDB` now tracks when a global schema lock corresponding to an exclusive lock on a tablespace is taken. If a different global schema lock request fails while the first lock, NDB assumes that there is a deadlock. In this case, the deadlock is handled by having the new request release all locks it previously acquired, then retrying them at a later point. (Bug #29394407)

  References: See also: Bug #29175268.
- **NDB Disk Data:** Following execution of `ALTER TABLESPACE`, SQL statements on an existing table using the affected tablespace failed with error 3508 `Dictionary object id (id) does not exist` where the object ID shown refers to the tablespace. Schema distribution of `ALTER TABLESPACE` involves dropping the old object from the data dictionary on a participating SQL node and creating a new one with a different dictionary object id, but the table object in the SQL node's data dictionary still used the old tablespace ID which rendered it unusable on the participants.

  To correct this problem, tables using the tablespace are now retrieved and stored prior to the creation of the new tablespace, and then Updated the new object ID of the tablespace after it has been created in the data dictionary. (Bug #29389168)

- **NDB Cluster APIs:** The memcached sources included with the NDB distribution would not build with `-Werror=format-security`. Now warnings are no longer treated as errors when compiling these files. (Bug #29512411)
- **NDB Cluster APIs:** It was not possible to scan a table whose `SingleUserMode` property had been set to `SingleUserModeReadWrite` or `SingleUserModeReadOnly`. (Bug #29493714)
- **NDB Cluster APIs:** The MGM API `ndb_logevent_get_next2()` function did not behave correctly on Windows and 32-bit Linux platforms. (Bug #94917, Bug #29609070)
- The version of Python expected by `ndb_setup.py` was not specified clearly on some platforms. (Bug #29818645)
- Lack of `SharedGlobalMemory` was incorrectly reported as lack of undo buffer memory, even though the cluster used no disk data tables. (Bug #29806771)

  References: This issue is a regression of: Bug #92125, Bug #28537319.
- Long `TCKEYREQ` signals did not always use the expected format when invoked from `TCINDXREQ` processing. (Bug #29772731)
- It was possible for an internal `NDB_SCHEMA_OBJECT` to be released too early or not at all; in addition, it was possible to create such an object that reused an existing key. (Bug #29759063)
- `ndb_restore` sometimes used `exit()` rather than `exitHandler()` to terminate the program, which could lead to resources not being properly freed. (Bug #29744353)
- Improved error message printed when the maximum offset for a `FIXED` column is exceeded. (Bug #29714670)
- Communication between the schema distribution client and the schema distribution coordinator is done using `NDB_SCHEMA_OBJECT` as well as by writing rows to the `ndb_schema` table in `NDB`. This allowed for the possibility of a number of different race conditions between when the registration of the schema operation and when the coordinator was notified of it.

  This fix addresses the following issues related to the situation just described:

  - The coordinator failed to abort active schema operations when the binary logging thread was restarted.
  - Schema operations already registered were not aborted properly.
  - The distribution client failed to detect correctly when schema distribution was not ready.
  - The distribution client, when killed, exited without marking the current schema operation as failed.
  - An operation in `NDB_SHARE` could be accessed without the proper locks being in place.

  In addition, usage of the `ndb_schema_share` global pointer was removed, and replaced with detecting whether the schema distribution is ready by checking whether an operation for `mysql.ndb_schema` has been created in `NDB_SHARE`. (Bug #29639381)
- With `DataMemory` set to 200 GB, `ndbmtd` failed to start. (Bug #29630367)
- When a backup fails due to `ABORT_BACKUP_ORD` being received while waiting for buffer space, the backup calls `closeScan()` and then sends a `SCAN_FRAGREQ` signal to the `DBLQH` block to close the scan. As part of receiving `SCAN_FRAGCONF` in response, `scanConf()` is called on the operation object for the file record which in turn calls `updateWritePtr()` on the file system buffer (`FsBuffer`). At this point the length sent by `updateWritePtr()` should be 0, but in this case was not, which meant that the buffer did not have enough space even though it did not, the problem being that the size is calculated as `scanStop - scanStart` and these values were held over since the previous `SCAN_FRAGCONF` was received, and were not reset due to being out of buffer space.

To avoid this problem, we now set `scanStart = scanStop` in `confirmBufferData()` (formerly `scanConfExtra()`) which is called as part of processing the `SCAN_FRAGCONF`, indirectly by `scanConf()` for the backup and first local checkpoint files, and directly for the LCP files which use only the operation record for the data buffer. (Bug #29601253)

- The setting for `MaxDMLOperationsPerTransaction` was not validated in a timely fashion, leading to data node failure rather than a management server error in the event that its value exceeded that of `MaxNoOfConcurrentOperations`. (Bug #29549572)
- Data nodes could fail due to an assert in the `DBTC` block under certain circumstances in resource-constrained environments. (Bug #29528188)
- An upgrade to NDB 7.6.9 or later from an earlier version could not be completed successfully if the redo log was filled to more than 25% of capacity. (Bug #29506844)
- When the `DBSPJ` block called the internal function `lookup_resume()` to schedule a previously enqueued operation, it used a correlation ID which could have been produced from its immediate ancestor in the execution order, and not its parent in the query tree as assumed. This could happen during execution of a `SELECT STRAIGHT_JOIN` query.

  Now `NDB` checks whether the execution ancestor is different from the query tree parent, and if not, performs a lookup of the query tree parent, and the parent's correlation ID is enqueued to be executed later. (Bug #29501263)
- When a new master took over, sending a `MASTER_LCP_REQ` signal and executing `MASTER_LCPCONF` from participating nodes, it expected that they had not completed the current local checkpoint under the previous master, which need not be true. (Bug #29487340, Bug #29601546)
- When restoring `TINYBLOB` columns, `ndb_restore` now treats them as having the `BINARY` character set. (Bug #29486538)
- When selecting a sorted result set from a query that included a `LIMIT` clause on a single table, and where the sort was executed as `Using filesort` and the `ref` access method was used on an ordered index, it was possible for the result set to be missing one or more rows. (Bug #29474188)
- Restoration of epochs by `ndb_restore` failed due to temporary redo errors. Now `ndb_restore` retries epoch updates when such errors occur. (Bug #29466089)
- `ndb_restore` tried to extract an 8-character substring of a table name when checking to determine whether or not the table was a blob table, regardless of the length of the name. (Bug #29465794)
- When a pushed join was used in combination with the `eq_ref` access method it was possible to obtain an incorrect join result due to the 1 row cache mechanism implemented in NDB 8.0.16 as part of the work done in that version to extend `NDB` condition pushdown by allowing referring values from previous tables. This issue is now fixed by turning off this caching mechanism and reading the row directly from the handler instead, when there is a pushed condition defined on the table. (Bug #29460314)
- Improved and made more efficient the conversion of rows by the `ha_ndbcluster` handler from the format used internally by `NDB` to that used by the MySQL server for columns that contain neither `BLOB` nor `BIT` values, which is the most common case. (Bug #29435461)
- A failed `DROP TABLE` could be attempted an infinite number of times in the event of a temporary error. Now in such cases, the number of retries is limited to 100. (Bug #29355155)
- `ndb_restore --restore-epoch` incorrectly reported the stop GCP as 1 less than the actual position. (Bug #29343655)
- A `SavedEvent` object in the `CMVMI` kernel block is written into a circular buffer. Such an object is split in two when wrapping at the end of the buffer; `NDB` looked beyond the end of the buffer instead of in the wrapped data at the buffer's beginning. (Bug #29336793)

- NDB did not compile with -DWITH_SYSTEM_LIBS=ON due to an incorrectly configured dependency on zlib. (Bug #29304517)
- Removed a memory leak found when running ndb_mgmd --config-file after compiling NDB with Clang 7. (Bug #29284643)
- Removed clang compiler warnings caused by usage of extra ; characters outside functions; these are incompatible with C++98. (Bug #29227925)
- Adding a column defined as TIMESTAMP DEFAULT CURRENT_TIMESTAMP to an NDB table is not supported with ALGORITHM=INPLACE. Attempting to do so now causes an error. (Bug #28128849)
- Added support which was missing in ndb_restore for conversions between the following sets of types:
  - BLOB and BINARY or VARBINARY columns
  - TEXT and BLOB columns
  - BLOB columns with unequal lengths
  - BINARY and VARBINARY columns with unequal lengths

  (Bug #28074988)
- Neither the MAX_EXECUTION_TIME optimizer hint nor the max_execution_time system variable was respected for DDL statements or queries against INFORMATION_SCHEMA tables while an NDB global schema lock was in effect. (Bug #27538139)
- DDL operations were not always performed correctly on database objects including databases and tables, when multi-byte character sets were used for the names of either or both of these. (Bug #27150334)
- ndb_import did not always free up all resources used before exiting. (Bug #27130143)
- NDBCLUSTER subscription log printouts provided only 2 words of the bitmap (in most cases containing 8 words), which made it difficult to diagnose schema distribution issues. (Bug #22180480)
- For certain tables with very large rows and a very large primary key, START BACKUP SNAPSHOTEND while performing inserts into one of these tables or START BACKUP SNAPSHOTSTART with concurrent deletes could lead to data node errors.

  As part of this fix, ndb_print_backup_file can now read backup files created in very old versions of NDB Cluster (6.3 and earlier); in addition, this utility can now also read undo log files. (Bug #94654, Bug #29485977)
- When one of multiple SQL nodes which were connected to the cluster was down and then rejoined the cluster, or a new SQL node joined the cluster, this node did not use the data dictionary correctly, and thus did not always add, alter, or drop databases properly when synchronizing with the existing SQL nodes.

  Now, during schema distribution at startup, the SQL node compares all databases on the data nodes with those in its own data dictionary. If any database on the data nodes is found to be missing from the SQL node's data dictionary, the SQL Node installs it locally using CREATE DATABASE; the database is created using the default MySQL Server database properties currently in effect on this SQL node. (WL #12731)

Changes in MySQL NDB Cluster 8.0.16 (2019-04-25, Development Milestone)

- Deprecation and Removal Notes

- SQL Syntax Notes
- Functionality Added or Changed
- Bugs Fixed

Deprecation and Removal Notes

- **Incompatible Change:** Distribution of privileges amongst MySQL servers connected to NDB Cluster, as implemented in NDB 7.6 and earlier, does not function in NDB 8.0, and most code supporting these has now been removed. When a `mysqld` detects such tables in `NDB`, it creates shadow tables local to itself using the `InnoDB` storage engine; these shadow tables are created on each MySQL server connected to an NDB cluster. Privilege tables using the `NDB` storage engine are not employed for access control; once all connected MySQL servers are upgraded, the privilege tables in `NDB` can be removed safely using `ndb_drop_table`.

  For compatibility reasons, `ndb_restore --restore-privilege-tables` can still be used to restore distributed privilege tables present in a backup taken from a previous release of NDB Cluster to a cluster running NDB 8.0. These tables are handled as described in the preceeding paragraph.

  For additional information regarding upgrades from previous NDB Cluster release series to NDB 8.0, see Upgrading and Downgrading NDB Cluster. (WL #12507, WL #12511)

SQL Syntax Notes

- **Incompatible Change:** For consistency with `InnoDB`, the `NDB` storage engine now uses a generated constraint name if the `CONSTRAINT symbol` clause is not specified, or the `CONSTRAINT` keyword is specified without a `symbol`. In previous `NDB` releases, `NDB` used the `FOREIGN KEY index_name` value.

  This change described above may introduce incompatibilities for applications that depend on the previous foreign key constraint naming behavior. (Bug #29173134)

Functionality Added or Changed

- **Packaging:** A Docker image for this release can be obtained from https://hub.docker.com/r/mysql/mysql-cluster/. (Bug #96084, Bug #30010921)
- Allocation of resources in the transaction corrdinator (see The DBTC Block) is now performed using dynamic memory pools. This means that resource allocation determined by data node configuration parameters such as those discussed in Transaction parameters and Transaction temporary storage is now limited so as not to exceed the total resources available to the transaction coordinator.

  As part of this work, several new data node parameters controlling transactional resources in `DBTC`, listed here, have also been added. For more information about these new parameters, see Transaction resource allocation parameters. (Bug #29164271, Bug #29194843, WL #9756, WL #12523)

  References: See also: Bug #29131828.

- `NDB` backups can now be performed in a parallel fashion on individual data nodes using multiple local data managers (LDMs). (Previously, backups were done in parallel across data nodes, but were always serial within data node processes.) No special syntax is required for the `START BACKUP` command in the `ndb_mgm` client to enable this feature, but all data nodes must be using multiple LDMs. This means that data nodes must be running `ndbmtd` and they must be configured to use multiple LDMs prior to taking the backup (see Multi-Threading Configuration Parameters (ndbmtd)).

  The `EnableMultithreadedBackup` data node parameter introduced in this release is enabled (set to 1) by default. You can disable multi-threaded backups and force the creation of single-threaded backups by setting this parameter to 0 on all data nodes or in the `[ndbd default]` section of the cluster's global configuration file (`config.ini`).

ndb_restore also now detects a multi-threaded backup and automatically attempts to restore it in parallel. It is also possible to restore backups taken in parallel to a previous version of NDB Cluster by slightly modifying the usual restore procedure.

For more information about taking and restoring NDB Cluster backups that were created using parallelism on the data nodes, see Taking an NDB Backup with Parallel Data Nodes, and Restoring from a backup taken in parallel. (Bug #28563639, Bug #28993400, WL #8517)

- The compile-cluster script included in the NDB source distribution no longer supports in-source builds. (WL #12303)
- Building with CMake3 is now supported by the compile-cluster script included in the NDB source distribution. (WL #12303)
- As part of its automatic synchronization mechanism, NDB now implements a metadata change monitor thread for detecting changes made to metadata for data objects such as tables, tablespaces, and log file groups with the MySQL data dictionary. This thread runs in the background, checking every 60 seconds for inconsistencies between the NDB dictionary and the MySQL data dictionary.

  The monitor polling interval can be adjusted by setting the value of the ndb_metadata_check_interval system variable, and can be disabled altogether by setting ndb_metadata_check to OFF. The number of times that inconsistencies have been detected since mysqld was last started is shown as the status variable, Ndb_metadata_detected_count. (WL #11913)
- Condition pushdown is no longer limited to predicate terms referring to column values from the same table to which the condition was being pushed; column values from tables earlier in the query plan can now also be referred to from pushed conditions. This lets the data nodes filter out more rows (in parallel), leaving less work to be performed by a single mysqld process, which is expected to provide significant improvements in query performance.

  For more information, see Engine Condition Pushdown Optimization. (WL #12686)

Bugs Fixed

- **Important Change; NDB Disk Data:** mysqldump terminated unexpectedly when attempting to dump NDB disk data tables. The underlying reason for this was that mysqldump expected to find information relating to undo log buffers in the EXTRA column of the INFORMATION_SCHEMA.FILES table but this information had been removed in NDB 8.0.13. This information is now restored to the EXTRA column. (Bug #28800252)
- **Important Change:** When restoring to a cluster using data node IDs different from those in the original cluster, ndb_restore tried to open files corresponding to node ID 0. To keep this from happening, the --nodeid and --backupid options—neither of which has a default value—are both now explicitly required when invoking ndb_restore. (Bug #28813708)
- **Important Change:** Starting with this release, the default value of the ndb_log_bin system variable is now FALSE. (Bug #27135706)
- **Packaging; MySQL NDB ClusterJ:** libndbclient was missing from builds on some platforms. (Bug #28997603)
- **NDB Disk Data:** When a log file group had more than 18 undo logs, it was not possible to restart the cluster. (Bug #251155785)

  References: See also: Bug #28922609.
- **NDB Disk Data:** Concurrent CREATE TABLE statements using tablespaces caused deadlocks between metadata locks. This occurred when Ndb_metadata_change_monitor acquired exclusive metadata locks on tablespaces and logfile groups after detecting metadata changes, due to the fact that each exclusive metadata lock in turn acquired a global schema lock. This fix

attempts to solve that issue by downgrading the locks taken by `Ndb_metadata_change_monitor` to `MDL_SHARED_READ`. (Bug #29175268)

References: See also: Bug #29394407.

- **NDB Disk Data:** The error message returned when validation of `MaxNoOfOpenFiles` in relation to `InitialNoOfOpenFiles` failed has been improved to make the nature of the problem clearer to users. (Bug #28943749)
- **NDB Disk Data:** Schema distribution of `ALTER TABLESPACE` and `ALTER LOGFILE GROUP` statements failed on a participant MySQL server if the referenced tablespace or log file group did not exist in its data dictionary. Now in such cases, the effects of the statement are distributed successfully regardless of any initial mismatch between MySQL servers. (Bug #28866336)
- **NDB Disk Data:** Repeated execution of `ALTER TABLESPACE ... ADD DATAFILE` against the same tablespace caused data nodes to hang and left them, after being killed manually, unable to restart. (Bug #22605467)
- **NDB Cluster APIs:** `NDB` now identifies short-lived transactions not needing the reduction of lock contention provided by `NdbBlob::close()` and no longer invokes this method in cases (such as when autocommit is enabled) in which unlocking merely causes extra work and round trips to be performed prior to committing or aborting the transaction. (Bug #29305592)

  References: See also: Bug #49190, Bug #11757181.
- **NDB Cluster APIs:** When the most recently failed operation was released, the pointer to it held by `NdbTransaction` became invalid and when accessed led to failure of the NDB API application. (Bug #29275244)
- **NDB Cluster APIs:** When the `NDB` kernel's `SUMA` block sends a `TE_ALTER` event, it does not keep track of when all fragments of the event are sent. When `NDB` receives the event, it buffers the fragments, and processes the event when all fragments have arrived. An issue could possibly arise for very large table definitions, when the time between transmission and reception could span multiple epochs; during this time, `SUMA` could send a `SUB_GCP_COMPLETE_REP` signal to indicate that it has sent all data for an epoch, even though in this case that is not entirely true since there may be fragments of a `TE_ALTER` event still waiting on the data node to be sent. Reception of the `SUB_GCP_COMPLETE_REP` leads to closing the buffers for that epoch. Thus, when `TE_ALTER` finally arrives, NDB assumes that it is a duplicate from an earlier epoch, and silently discards it.

  We fix the problem by making sure that the `SUMA` kernel block never sends a `SUB_GCP_COMPLETE_REP` for any epoch in which there are unsent fragments for a `SUB_TABLE_DATA` signal.

  This issue could have an impact on NDB API applications making use of `TE_ALTER` events. (SQL nodes do not make any use of `TE_ALTER` events and so they and applications using them were not affected.) (Bug #28836474)
- When a pushed join executing in the `DBSPJ` block had to store correlation IDs during query execution, memory for these was allocated for the lifetime of the entire query execution, even though these specific correlation IDs are required only when producing the most recent batch in the result set. Subsequent batches require additional correlation IDs to be stored and allocated; thus, if the query took sufficiently long to complete, this led to exhaustion of query memory (error 20008). Now in such cases, memory is allocated only for the lifetime of the current result batch, and is freed and made available for re-use following completion of the batch. (Bug #29336777)

  References: See also: Bug #26995027.
- When comparing or hashing a fixed-length string that used a `NO_PAD` collation, any trailing padding characters (typically spaces) were sent to the hashing and comparison functions such that they became significant, even though they were not supposed to be. Now any such trailing spaces are trimmed from a fixed-length string whenever a `NO_PAD` collation is specified.

> **Note**
>
> Since NO_PAD collations were introduced as part of UCA-9.0 collations in MySQL 8.0, there should be no impact relating to this fix on upgrades to NDB 8.0 from previous GA releases of NDB Cluster.

(Bug #29322313)

- When a NOT IN or NOT BETWEEN predicate was evaluated as a pushed condition, NULL values were not eliminated by the condition as specified in the SQL standard. (Bug #29232744)

  References: See also: Bug #28672214.

- Internally, NDB treats NULL as less than any other value, and predicates of the form *column < value* or *column <= value* are checked for possible nulls. Predicates of the form *value > column* or *value >= column* were not checked, which could lead to errors. Now in such cases, these predicates are rewritten so that the column comes first, so that they are also checked for the presence of NULL. (Bug #29231709)

  References: See also: Bug #92407, Bug #28643463.

- After folding of constants was implemented in the MySQL Optimizer, a condition containing a DATE or DATETIME literal could no longer be pushed down by NDB. (Bug #29161281)

- When a join condition made a comparison between a column of a temporal data type such as DATE or DATETIME and a constant of the same type, the predicate was pushed if the condition was expressed in the form *column operator constant*, but not when in inverted order (as *constant inverse_operator column*). (Bug #29058732)

- When processing a pushed condition, NDB did not detect errors or warnings thrown when a literal value being compared was outside the range of the data type it was being compared with,and thus truncated. This could lead to excess or missing rows in the result. (Bug #29054626)

- If an EQ_REF or REF key in the child of a pushed join referred to any columns of a table not a member of the pushed join, this table was not an NDB table (because its format was of nonnative endianness), and the data type of the column being joined on was stored in an endian-sensitive format, then the key generated was generated, likely resulting in the return of an (invalid) empty join result.

  Since only big endian platforms may store tables in nonnative (little endian) formats, this issue was expected only on such platforms, most notably SPARC, and not on x86 platforms. (Bug #29010641)

- API and data nodes running NDB 7.6 and later could not use an existing parsed configuration from an earlier release series due to being overly strict with regard to having values defined for configuration parameters new to the later release, which placed a restriction on possible upgrade paths. Now NDB 7.6 and later are less strict about having all new parameters specified explicitly in the configuration which they are served, and use hard-coded default values in such cases. (Bug #28993400)

- NDB 7.6 SQL nodes hung when trying to connect to an NDB 8.0 cluster. (Bug #28985685)

- The schema distribution data maintained in the NDB binary logging thread keeping track of the number of subscribers to the NDB schema table always allocated some memory structures for 256 data nodes regardless of the actual number of nodes. Now NDB allocates only as many of these structures as are actually needed. (Bug #28949523)

- Added DUMP 406 (NdbfsDumpRequests) to provide NDB file system information to global checkpoint and local checkpoint stall reports in the node logs. (Bug #28922609)

- When a joined table was eliminated early as not pushable, it could not be referred to in any subsequent join conditions from other tables without eliminating those conditions from consideration even if those conditions were otherwise pushable. (Bug #28898811)

- When starting or restarting an SQL node and connecting to a cluster where `NDB` was already started, `NDB` reported Error 4009 `Cluster Failure` because it could not acquire a global schema lock. This was because the MySQL Server as part of initialization acquires exclusive metadata locks in order to modify internal data structures, and the `ndbcluster` plugin acquires the global schema lock. If the connection to `NDB` was not yet properly set up during `mysqld` initialization, `mysqld` received a warning from `ndbcluster` when the latter failed to acquire global schema lock, and printed it to the log file, causing an unexpected error in the log. This is fixed by not pushing any warnings to background threads when failure to acquire a global schema lock occurs and pushing the `NDB` error as a warning instead. (Bug #28898544)

- A race condition between the `DBACC` and `DBLQH` kernel blocks occurred when different operations in a transaction on the same row were concurrently being prepared and aborted. This could result in `DBTUP` attempting to prepare an operation when a preceding operation had been aborted, which was unexpected and could thus lead to undefined behavior including potential data node failures. To solve this issue, `DBACC` and `DBLQH` now check that all dependencies are still valid before attempting to prepare an operation.

  Note

  This fix also supersedes a previous one made for a related issue which was originally reported as Bug #28500861.

  (Bug #28893633)

- Where a data node was restarted after a configuration change whose result was a decrease in the sum of `MaxNoOfTables`, `MaxNoOfOrderedIndexes`, and `MaxNoOfUniqueHashIndexes`, it sometimes failed with a misleading error message which suggested both a temporary error and a bug, neither of which was the case.

  The failure itself is expected, being due to the fact that there is at least one table object with an ID greater than the (new) sum of the parameters just mentioned, and that this table cannot be restored since the maximum value for the ID allowed is limited by that sum. The error message has been changed to reflect this, and now indicates that this is a permanent error due to a problem configuration. (Bug #28884880)

- The `ndbinfo.cpustat` table reported inaccurate information regarding send threads. (Bug #28884157)

- Execution of an LCP_COMPLETE_REP signal from the master while the LCP status was IDLE led to an assertion. (Bug #28871889)

- `NDB` now provides on-the-fly `.frm` file translation during discovery of tables created in versions of the software that did not support the MySQL Data Dictionary. Previously, such translation of tables that had old-style metadata was supported only during schema synchronization during MySQL server startup, but not subsequently, which led to errors when `NDB` tables having old-style metadata, created by `ndb_restore` and other such tools after `mysqld` had been started, were accessed using `SHOW CREATE TABLE` or `SELECT`; these tables were usable only after restarting `mysqld`. With this fix, the restart is no longer required. (Bug #28841009)

- An in-place upgrade to an NDB 8.0 release from an earlier relase did not remove `.ndb` files, even though these are no longer used in NDB 8.0. (Bug #28832816)

- Removed `storage/ndb/demos` and the demonstration scripts and support files it contained from the source tree. These were obsolete and unmaintained, and did not function with any current version of NDB Cluster.

  Also removed `storage/ndb/include/newtonapi`, which included files relating to an obsolete and unmaintained API not supported in any release of NDB Cluster, as well as references elsewhere to these files. (Bug #28808766)

- There was no version compatibility table for NDB 8.x; this meant that API nodes running NDB 8.0.13 or 7.6.x could not connect to data nodes running NDB 8.0.14. This issue manifested itself for NDB API users as a failure in `wait_until_ready()`. (Bug #28776365)

  References: See also: Bug #18886034, Bug #18874849.

- Issuing a `STOP` command in the `ndb_mgm` client caused `ndbmtd` processes which had recently been added to the cluster to hang in Phase 4 during shutdown. (Bug #28772867)

- A fix for a previous issue disabled the usage of pushed conditions for lookup type (`eq_ref`) operations in pushed joins. It was thought at the time that not pushing a lookup condition would not have any measurable impact on performance, since only a single row could be eliminated if the condition failed. The solution implemented at that time did not take into account the possibility that, in a pushed join, a lookup operation could be a parent operation for other lookups, and even scan operations, which meant that eliminating a single row could actually result in an entire branch being eliminated in error. (Bug #28728603)

  References: This issue is a regression of: Bug #27397802.

- When a local checkpoint (LCP) was complete on all data nodes except one, and this node failed, `NDB` did not continue with the steps required to finish the LCP. This led to the following issues:

  No new LCPs could be started.

  Redo and Undo logs were not trimmed and so grew excessively large, causing an increase in times for recovery from disk. This led to write service failure, which eventually led to cluster shutdown when the head of the redo log met the tail. This placed a limit on cluster uptime.

  Node restarts were no longer possible, due to the fact that a data node restart requires that the node's state be made durable on disk before it can provide redundancy when joining the cluster. For a cluster with two data nodes and two fragment replicas, this meant that a restart of the entire cluster (system restart) was required to fix the issue (this was not necessary for a cluster with two fragment replicas and four or more data nodes). (Bug #28728485, Bug #28698831)

  References: See also: Bug #11757421.

- The pushability of a condition to `NDB` was limited in that all predicates joined by a logical `AND` within a given condition had to be pushable to `NDB` in order for the entire condition to be pushed. In some cases this severely restricted the pushability of conditions. This fix breaks up the condition into its components, and evaluates the pushability of each predicate; if some of the predicates cannot be pushed, they are returned as a remainder condition which can be evaluated by the MySQL server. (Bug #28728007)

- Running `ANALYZE TABLE` on an `NDB` table with an index having longer than the supported maximum length caused data nodes to fail. (Bug #28714864)

- It was possible in certain cases for nodes to hang during an initial restart. (Bug #28698831)

  References: See also: Bug #27622643.

- When a condition was pushed to a storage engine, it was re-evaluated by the server, in spite of the fact that only rows matching the pushed condition should ever be returned to the server in such cases. (Bug #28672214)

- In some cases, one and sometimes more data nodes underwent an unplanned shutdown while running `ndb_restore`. This occurred most often, but was not always restircted to, when restoring to a cluster having a different number of data nodes from the cluster on which the original backup had been taken.

  The root cause of this issue was exhaustion of the pool of `SafeCounter` objects, used by the `DBDICT` kernel block as part of executing schema transactions, and taken from a per-block-instance pool shared with protocols used for `NDB` event setup and subscription processing. The concurrency of event setup and subscription processing is such that the `SafeCounter` pool can be exhausted; event and subscription processing can handle pool exhaustion, but schema transaction processing could not, which could result in the node shutdown experienced during restoration.

  This problem is solved by giving `DBDICT` schema transactions an isolated pool of reserved `SafeCounters` which cannot be exhausted by concurrent `NDB` event activity. (Bug #28595915)

- When a backup aborted due to buffer exhaustion, synchronization of the signal queues prior to the expected drop of triggers for insert, update, and delete operations resulted in abort signals being processed before the `STOP_BACKUP` phase could continue. The abort changed the backup status to `ABORT_BACKUP_ORD`, which led to an unplanned shutdown of the data node since resuming `STOP_BACKUP` requires that the state be `STOP_BACKUP_REQ`. Now the backup status is not set to `STOP_BACKUP_REQ` (requesting the backup to continue) until after signal queue synchronization is complete. (Bug #28563639)

- The output of `ndb_config --configinfo --xml --query-all` now shows that configuration changes for the `ThreadConfig` and `MaxNoOfExecutionThreads` data node parameters require system initial restarts (`restart="system" initial="true"`). (Bug #28494286)

- After a commit failed due to an error, `mysqld` shut down unexpectedly while trying to get the name of the table involved. This was due to an issue in the internal function `ndbcluster_print_error()`. (Bug #28435082)

- API nodes should observe that a node is moving through `SL_STOPPING` phases (graceful stop) and stop using the node for new transactions, which minimizes potential disruption in the later phases of the node shutdown process. API nodes were only informed of node state changes via periodic heartbeat signals, and so might not be able to avoid interacting with the node shutting down. This generated unnecessary failures when the heartbeat interval was long. Now when a data node is being gracefully stopped, all API nodes are notified directly, allowing them to experience minimal disruption. (Bug #28380808)

- `ndb_config --diff-default` failed when trying to read a parameter whose default value was the empty string (`""`). (Bug #27972537)

- `ndb_restore` did not restore autoincrement values correctly when one or more staging tables were in use. As part of this fix, we also in such cases block applying of the `SYSTAB_0` backup log, whose content continued to be applied directly based on the table ID, which could ovewrite the autoincrement values stored in `SYSTAB_0` for unrelated tables. (Bug #27917769, Bug #27831990)

  References: See also: Bug #27832033.

- `ndb_restore` employed a mechanism for restoring autoincrement values which was not atomic, and thus could yield incorrect autoincrement values being restored when multiple instances of `ndb_restore` were used in parallel. (Bug #27832033)

  References: See also: Bug #27917769, Bug #27831990.

- Executing `SELECT * FROM INFORMATION_SCHEMA.TABLES` caused SQL nodes to restart in some cases. (Bug #27613173)

- When tables with `BLOB` columns were dropped and then re-created with a different number of `BLOB` columns the event definitions for monitoring table changes could become inconsistent in certain error situations involving communication errors when the expected cleanup of the corresponding events

was not performed. In particular, when the new versions of the tables had more BLOB columns than the original tables, some events could be missing. (Bug #27072756)

- When query memory was exhausted in the DBSPJ kernel block while storing correlation IDs for deferred operations, the query was aborted with error status 20000 Query aborted due to out of query memory. (Bug #26995027)

  References: See also: Bug #86537.

- When running a cluster with 4 or more data nodes under very high loads, data nodes could sometimes fail with Error 899 Rowid already allocated. (Bug #25960230)

- mysqld shut down unexpectedly when a purge of the binary log was requested before the server had completely started, and it was thus not yet ready to delete rows from the ndb_binlog_index table. Now when this occurs, requests for any needed purges of the ndb_binlog_index table are saved in a queue and held for execution when the server has completely started. (Bug #25817834)

- MaxBufferedEpochs is used on data nodes to avoid excessive buffering of row changes due to lagging NDB event API subscribers; when epoch acknowledgements from one or more subscribers lag by this number of epochs, an asynchronous disconnection is triggered, allowing the data node to release the buffer space used for subscriptions. Since this disconnection is asynchronous, it may be the case that it has not completed before additional new epochs are completed on the data node, resulting in new epochs not being able to seize GCP completion records, generating warnings such as those shown here:

  ```
  [ndbd] ERROR    -- c_gcp_list.seize() failed...

  ...

  [ndbd] WARNING  -- ACK wo/ gcp record...
  ```

  And leading to the following warning:

  ```
  Disconnecting node %u because it has exceeded MaxBufferedEpochs
  (100 > 100), epoch ....
  ```

  This fix performs the following modifications:

  - Modifies the size of the GCP completion record pool to ensure that there is always some extra headroom to account for the asynchronous nature of the disconnect processing previously described, thus avoiding c_gcp_list seize failures.

  - Modifies the wording of the MaxBufferedEpochs warning to avoid the contradictory phrase “100 > 100”.

  (Bug #20344149)

- Asynchronous disconnection of mysqld from the cluster caused any subsequent attempt to start an NDB API transaction to fail. If this occurred during a bulk delete operation, the SQL layer called HA::end_bulk_delete(), whose implementation by ha_ndbcluster assumed that a transaction had been started, and could fail if this was not the case. This problem is fixed by checking that the transaction pointer used by this method is set before referencing it. (Bug #20116393)

- Removed warnings raised when compiling NDB with Clang 6. (Bug #93634, Bug #29112560)

- When executing the redo log in debug mode it was possible for a data node to fail when deallocating a row. (Bug #93273, Bug #28955797)

- An NDB table having both a foreign key on another NDB table using ON DELETE CASCADE and one or more TEXT or BLOB columns leaked memory.

  As part of this fix, ON DELETE CASCADE is no longer supported for foreign keys on NDB tables when the child table contains a column that uses any of the BLOB or TEXT types. (Bug #89511, Bug #27484882)

Changes in MySQL NDB Cluster 8.0.14 (2019-01-21, Development Milestone)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Performance:** This release introduces a number of significant improvements in the performance of scans; these are listed here:

  - Row checksums help detect hardware issues, but do so at the expense of performance. NDB now offers the possibility of disabling these by setting the new ndb_row_checksum server system variable to 0; doing this means that row checksums are not used for new or altered tables. This can have a significant impact (5 to 10 percent, in some cases) on performance for all types of queries. This variable is set to 1 by default, to provide compatibility with the previous behavior.
  - A query consisting of a scan can execute for a longer time in the LDM threads when the queue is not busy.
  - Previously, columns were read before checking a pushed condition; now checking of a pushed condition is done before reading any columns.
  - Performance of pushed joins should see significant improvement when using range scans as part of join execution.

  (WL #11722)

- **NDB Disk Data:** NDB now implements schema distribution of disk data objects including tablespaces and log file groups by SQL nodes when they connect to a cluster, just as it does for NDB databases and in-memory tables. This eliminates a possible mismatch between the MySQL data dictionary and the NDB dictionary following a native backup and restore that could arise when disk data tablespaces and undo log file groups were restored to the NDB dictionary, but not to the MySQL Server's data dictionary. (WL #12172)

- **NDB Disk Data:** NDB now makes use of the MySQL data dictionary to ensure correct distribution of tablespaces and log file groups across all cluster SQL nodes when connecting to the cluster. (WL #12333)

- The extra metadata property for NDB tables is now used to store information from the MySQL data dictionary. Because this information is significantly larger than the binary representation previously stored here (a .frm file, no longer used), the hard-coded size limit for this extra metadata has been increased.

  This change can have an impact on downgrades: Trying to read NDB tables created in NDB 8.0.14 and later may cause data nodes running NDB 8.0.13 or earlier to fail on startup with NDB error code 2355 Failure to restore schema: Permanent error, external action needed: Resource configuration error. This can happen if the table's metadata exceeds 6K in size, which was the old limit. Tables created in NDB 8.0.13 and earlier can be read by later versions without any issues.

  For more information, see Changes in NDB table extra metadata, and See also MySQL Data Dictionary. (Bug #27230681, WL #10665)

Bugs Fixed

- **Packaging:** Expected NDB header files were in the `devel` RPM package instead of `libndbclient-devel`. (Bug #84580, Bug #26448330)
- **ndbmemcache:** `libndbclient.so` was not able to find and load `libssl.so`, which could cause issues with `ndbmemcache` and Java-based programs using `NDB`. (Bug #26824659)

  References: See also: Bug #27882088, Bug #28410275.
- **MySQL NDB ClusterJ:** The `ndb.clusterj` test for NDB 8.0.13 failed when being run more than once. This was deal to a new, stricter rule with NDB 8.0.13 that did not allow temporary files being left behind in the variable folder of `mysql-test-run (mtr)`. With this fix, the temporary files are deleted before the test is executed. (Bug #28279038)
- **MySQL NDB ClusterJ:** A `NullPointerException` was thrown when a full table scan was performed with ClusterJ on tables containing either a BLOB or a TEXT field. It was because the proper object initializations were omitted, and they have now been added by this fix. (Bug #28199372, Bug #91242)
- The `version_comment` system variable was not correctly configured in `mysqld` binaries and returned a generic pattern instead of the proper value. This affected all NDB Cluster binary releases with the exception of `.deb` packages. (Bug #29054235)
- Trying to build from source using `-DWITH_NDBCLUSTER` and `-Werror` failed with GCC 8. (Bug #28707282)
- When copying deleted rows from a live node to a node just starting, it is possible for one or more of these rows to have a global checkpoint index equal to zero. If this happened at the same time that a full local checkpoint was started due to the undo log getting full, the `LCP_SKIP` bit was set for a row having GCI = 0, leading to an unplanned shutdown of the data node. (Bug #28372628)
- `ndbmtd` sometimes experienced a hang when exiting due to log thread shutdown. (Bug #28027150)
- `NDB` has an upper limit of 128 characters for a fully qualified table name. Due to the fact that `mysqld` names `NDB` tables using the format *`database_name/catalog_name/table_name`*, where *`catalog_name`* is always `def`, it is possible for statements such as `CREATE TABLE` to fail in spite of the fact that neither the table name nor the database name exceeds the 63-character limit imposed by `NDB`. The error raised in such cases was misleading and has been replaced. (Bug #27769521)

  References: See also: Bug #27769801.
- When the `SUMA` kernel block receives a `SUB_STOP_REQ` signal, it executes the signal then replies with `SUB_STOP_CONF`. (After this response is relayed back to the API, the API is open to send more `SUB_STOP_REQ` signals.) After sending the `SUB_STOP_CONF`, SUMA drops the subscription if no subscribers are present, which involves sending multiple `DROP_TRIG_IMPL_REQ` messages to `DBTUP`. LocalProxy can handle up to 21 of these requests in parallel; any more than this are queued in the Short Time Queue. When execution of a `DROP_TRIG_IMPL_REQ` was delayed, there was a chance for the queue to become overloaded, leading to a data node shutdown with `Error in short time queue`.

  This issue is fixed by delaying the execution of the `SUB_STOP_REQ` signal if `DBTUP` is already handling `DROP_TRIG_IMPL_REQ` signals at full capacity, rather than queueing up the `DROP_TRIG_IMPL_REQ` signals. (Bug #26574003)
- `ndb_restore` returned -1 instead of the expected exit code in the event of an index rebuild failure. (Bug #25112726)
- When starting, a data node copies metadata, while a local checkpoint updates metadata. To avoid any conflict, any ongoing LCP activity is paused while metadata is being copied. An issue arose when a local checkpoint was paused on a given node, and another node that was also restarting checked for a complete LCP on this node; the check actually caused the LCP to be completed before

copying of metadata was complete and so ended the pause prematurely. Now in such cases, the LCP completion check waits to complete a paused LCP until copying of metadata is finished and the pause ends as expected, within the LCP in which it began. (Bug #24827685)

- `ndbout` and `ndberr` became invalid after exiting from `mgmd_run()`, and redirecting to them before the next call to `mgmd_run()` caused a segmentation fault, during an `ndb_mgmd` service restart. This fix ensures that `ndbout` and `ndberr` remain valid at all times. (Bug #17732772, Bug #28536919)

- `NdbScanFilter` did not always handle `NULL` according to the SQL standard, which could result in sending non-qualifying rows to be filtered (otherwise not necessary) by the MySQL server. (Bug #92407, Bug #28643463)

  References: See also: Bug #93977, Bug #29231709.

- The internal function `ndb_my_error()` was used in `ndbcluster_get_tablespace_statistics()` and `prepare_inplace_alter_table()` to report errors when the function failed to interact with `NDB`. The function was expected to push the NDB error as warning on the stack and then set an error by translating the NDB error to a MySQL error and then finally call `my_error()` with the translated error. When calling `my_error()`, the function extracts a format string that may contain placeholders and use the format string in a function similar to `sprintf()`, which in this case could read arbitrary memory leading to a segmentation fault, due to the fact that `my_error()` was called without any arguments.

  The fix is always to push the NDB error as a warning and then set an error with a provided message. A new helper function has been added to `Thd_ndb` to be used in place of `ndb_my_error()`. (Bug #92244, Bug #28575934)

- Running out of undo log buffer memory was reported using error 921 `Out of transaction memory ... (increase SharedGlobalMemory)`.

  This problem is fixed by introducing a new error code 923 `Out of undo buffer memory (increase UNDO_BUFFER_SIZE)`. (Bug #92125, Bug #28537319)

- When moving an `OperationRec` from the serial to the parallel queue, `Dbacc::startNext()` failed to update the `Operationrec::OP_ACC_LOCK_MODE` flag which is required to reflect the accumulated `OP_LOCK_MODE` of all previous operations in the parallel queue. This inconsistency in the ACC lock queues caused the scan lock takeover mechanism to fail, as it incorrectly concluded that a lock to take over was not held. The same failure caused an assert when aborting an operation that was a member of such an inconsistent parallel lock queue. (Bug #92100, Bug #28530928)

- `ndb_restore` did not free all memory used after being called to restore a table that already existed. (Bug #92085, Bug #28525898)

- A data node failed during startup due to the arrival of a `SCAN_FRAGREQ` signal during the restore phase. This signal originated from a scan begun before the node had previously failed and which should have been aborted due to the involvement of the failed node in it. (Bug #92059, Bug #28518448)

- `DBTUP` sent the error `Tuple corruption detected` when a read operation attempted to read the value of a tuple inserted within the same transaction. (Bug #92009, Bug #28500861)

  References: See also: Bug #28893633.

- False constraint violation errors could occur when executing updates on self-referential foreign keys. (Bug #91965, Bug #28486390)

  References: See also: Bug #90644, Bug #27930382.

- An `NDB` internal trigger definition could be dropped while pending instances of the trigger remained to be executed, by attempting to look up the definition for a trigger which had already been released. This caused unpredictable and thus unsafe behavior possibly leading to data node failure. The root cause of the issue lay in an invalid assumption in the code relating to determining whether a given

trigger had been released; the issue is fixed by ensuring that the behavior of NDB, when a trigger definition is determined to have been released, is consistent, and that it meets expectations. (Bug #91894, Bug #28451957)

- In some cases, a workload that included a high number of concurrent inserts caused data node failures when using debug builds. (Bug #91764, Bug #28387450, Bug #29055038)
- During an initial node restart with disk data tables present and TwoPassInitialNodeRestartCopy enabled, DBTUP used an unsafe scan in disk order. Such scans are no longer employed in this case. (Bug #91724, Bug #28378227)
- Checking for old LCP files tested the table version, but this was not always dependable. Now, instead of relying on the table version, the check regards as invalid any LCP file having a maxGCI smaller than its createGci. (Bug #91637, Bug #28346565)
- In certain cases, a cascade update trigger was fired repeatedly on the same record, which eventually consumed all available concurrent operations, leading to Error 233 Out of operation records in transaction coordinator (increase MaxNoOfConcurrentOperations). If MaxNoOfConcurrentOperations was set to a value sufficiently high to avoid this, the issue manifested as data nodes consuming very large amounts of CPU, very likely eventually leading to a timeout. (Bug #91472, Bug #28262259)

Changes in MySQL NDB Cluster 8.0.13 (2018-10-23, Development Milestone)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change; NDB Disk Data:** The following changes are made in the display of information about Disk Data files in the INFORMATION_SCHEMA.FILES table:
  - Tablespaces and log file groups are no longer represented in the FILES table. (These constructs are not actually files.)
  - Each data file is now represented by a single row in the FILES table. Each undo log file is also now represented in this table by one row only. (Previously, a row was displayed for each copy of each of these files on each data node.)
  - For rows corresponding to data files or undo log files, node ID and undo log buffer information is no longer displayed in the EXTRA column of the FILES table.

    Important

    The removal of undo log buffer information is reverted in NDB 8.0.15. (Bug #92796, Bug #28800252)

  (WL #11553)
- **Important Change; NDB Client Programs:** Removed the deprecated --ndb option for perror. Use ndb_perror to obtain error message information from NDB error codes instead. (Bug #81705, Bug #23523957)

  References: See also: Bug #81704, Bug #23523926.
- **Important Change:** Beginning with this release, MySQL NDB Cluster is being developed in parallel with the standard MySQL 8.0 server under a new unified release model with the following features:
  - NDB 8.0 is developed in, built from, and released with the MySQL 8.0 source code tree.

- The numbering scheme for NDB Cluster 8.0 releases follows the scheme for MySQL 8.0, starting with the current MySQL release (8.0.13).
- Building the source with NDB support appends `-cluster` to the version string returned by `mysql -V`, as shown here:

  ```
  shell≫ mysql -V
  mysql  Ver 8.0.13-cluster for Linux on x86_64 (Source distribution)
  ```

  NDB binaries continue to display both the MySQL Server version and the NDB engine version, like this:

  ```
  shell> ndb_mgm -V
  MySQL distrib mysql-8.0.13 ndb-8.0.13-dmr, for Linux (x86_64)
  ```

  In MySQL Cluster NDB 8.0, these two version numbers are always the same.

To build the MySQL 8.0.13 (or later) source with NDB Cluster support, use the CMake option `-DWITH_NDBCLUSTER`. (WL #11762)

- **NDB Cluster APIs:** Added the `Table` methods `getExtraMetadata()` and `setExtraMetadata()`. (WL #9865)
- `INFORMATION_SCHEMA` tables now are populated with tablespace statistics for MySQL Cluster tables. (Bug #27167728)
- It is now possible to specify a set of cores to be used for I/O threads performing offline multithreaded builds of ordered indexes, as opposed to normal I/O duties such as file I/O , compression , or decompression. “Offline” in this context refers to building of ordered indexes performed when the parent table is not being written to; such building takes place when an `NDB` cluster performs a node or system restart, or as part of restoring a cluster from backup using `ndb_restore --rebuild-indexes`.

  In addition, the default behaviour for offline index build work is modified to use all cores available to `ndbmtd`, rather limiting itself to the core reserved for the I/O thread. Doing so can improve restart and restore times and performance, availability, and the user experience.

  This enhancement is implemented as follows:

  1. The default value for `BuildIndexThreads` is changed from 0 to 128. This means that offline ordered index builds are now multithreaded by default.
  2. The default value for `TwoPassInitialNodeRestartCopy` is changed from `false` to `true`. This means that an initial node restart first copies all data from a “live” node to one that is starting —without creating any indexes—builds ordered indexes offline, and then again synchronizes its data with the live node, that is, synchronizing twice and building indexes offline between the two synchonizations. This causes an initial node restart to behave more like the normal restart of a node, and reduces the time required for building indexes.
  3. A new thread type (`idxbld`) is defined for the `ThreadConfig` configuration parameter, to allow locking of offline index build threads to specific CPUs.

  In addition, `NDB` now distinguishes the thread types that are accessible to “ThreadConfig” by the following two criteria:

  1. Whether the thread is an execution thread. Threads of types `main`, `ldm`, `recv`, `rep`, `tc`, and `send` are execution threads; thread types `io`, `watchdog`, and `idxbld` are not.

2. Whether the allocation of the thread to a given task is permanent or temporary. Currently all thread types except `idxbld` are permanent.

For additonal information, see the descriptions of the parameters in the Manual. (Bug #25835748, Bug #26928111)

- When performing an NDB backup, the `ndbinfo.logbuffers` table now displays information regarding buffer usage by the backup process on each data node. This is implemented as rows reflecting two new log types in addition to `REDO` and `DD-UNDO`. One of these rows has the log type `BACKUP-DATA`, which shows the amount of data buffer used during backup to copy fragments to backup files. The other row has the log type `BACKUP-LOG`, which displays the amount of log buffer used during the backup to record changes made after the backup has started. One each of these `log_type` rows is shown in the `logbuffers` table for each data node in the cluster. Rows having these two log types are present in the table only while an NDB backup is currently in progress. (Bug #25822988)

- Added the `ODirectSyncFlag` configuration parameter for data nodes. When enabled, the data node treats all completed filesystem writes to the redo log as though they had been performed using `fsync`.

  Note

  This parameter has no effect if at least one of the following conditions is true:

  - `ODirect` is not enabled.
  - `InitFragmentLogFiles` is set to `SPARSE`.

  (Bug #25428560)

- Added the `--logbuffer-size` option for `ndbd` and `ndbmtd`, for use in debugging with a large number of log messages. This controls the size of the data node log buffer; the default (32K) is intended for normal operations. (Bug #89679, Bug #27550943)

- Prior to NDB 8.0, all string hashing was based on first transforming the string into a normalized form, then MD5-hashing the resulting binary image. This could give rise to some performance problems, for the following reasons:

  - The normalized string is always space padded to its full length. For a `VARCHAR`, this often involved adding more spaces than there were characters in the original string.
  - The string libraries were not optimized for this space padding, and added considerable overhead in some use cases.
  - The padding semantics varied between character sets, some of which were not padded to their full length.
  - The transformed string could become quite large, even without space padding; some Unicode 9.0 collations can transform a single code point into 100 bytes of character data or more.
  - Subsequent MD5 hashing consisted mainly of padding with spaces, and was not particularly efficient, possibly causing additional performance penalties by flush significant portions of the L1 cache.

  Collations provide their own hash functions, which hash the string directly without first creating a normalized string. In addition, for Unicode 9.0 collations, the hashes are computed without padding.

NDB now takes advantage of this built-in function whenever hashing a string identified as using a Unicode 9.0 collation.

Since, for other collations there are existing databases which are hash partitioned on the transformed string, NDB continues to employ the previous method for hashing strings that use these, to maintain compatibility. (Bug #89609, Bug #27523758)

References: See also: Bug #89590, Bug #27515000, Bug #89604, Bug #27522732.

- A table created in NDB 7.6 and earlier contains metadata in the form of a compressed .frm file, which is no longer supported in MySQL 8.0. To facilitate online upgrades to NDB 8.0, NDB performs on-the-fly translation of this metadata and writes it into the MySQL Server's data dictionary, which enables the mysqld in NDB Cluster 8.0 to work with the table without preventing subsequent use of the table by a previous version of the NDB software.

  Important

  Once a table's structure has been modified in NDB 8.0, its metadata is stored using the Data Dictionary, and it can no longer be accessed by NDB 7.6 and earlier.

  This enhancement also makes it possible to restore an NDB backup made using an earlier version to a cluster running NDB 8.0 (or later). (WL #10167)

Bugs Fixed

- **Important Change; NDB Disk Data:** It was possible to issue a CREATE TABLE statement that referred to a nonexistent tablespace. Now such a statement fails with an error. (Bug #85859, Bug #25860404)

- **Important Change; NDB Replication:** Because the MySQL Server now executes RESET MASTER with a global read lock, the behavior of this statement when used with NDB Cluster has changed in the following two respects:

  - It is no longer guaranteed to be synchronous; that is, it is now possible that a read coming immediately before RESET MASTER is issued may not be logged until after the binary log has been rotated.

  - It now behaves identically, regardless of whether the statement is issued on the same SQL node that is writing the binary log, or on a different SQL node in the same cluster.

  Note

  SHOW BINLOG EVENTS, FLUSH LOGS, and most data definition statements continue, as they did in previous NDB versions, to operate in a synchronous fashion.

  (Bug #89976, Bug #27665565)

- **Important Change:** NDB supports any of the following three values for the CREATE TABLE statement's ROW_FORMAT option: DEFAULT, FIXED, and DYNAMIC. Formerly, any values other than these were accepted but resulted in DYNAMIC being used. Now a CREATE TABLE statement that attempts to create an NDB table fails with an error if ROW_FORMAT is used, and does not have one of the three values listed. (Bug #88803, Bug #27230898)

- **Microsoft Windows; ndbinfo Information Database:** The process ID of the monitor process used on Windows platforms by RESTART to spawn and restart a mysqld is now shown in the ndbinfo.processes table as an angel_pid. (Bug #90235, Bug #27767237)

- **NDB Cluster APIs:** The example NDB API programs ndbapi_array_simple and ndbapi_array_using_adapter did not perform cleanup following execution; in addition, the

example program `ndbapi_simple_dual` did not check to see whether the table used by this example already existed. Due to these issues, none of these examples could be run more than once in succession.

The issues just described have been corrected in the example sources, and the relevant code listings in the NDB API documentation have been updated to match. (Bug #27009386)

- **NDB Cluster APIs:** A previous fix for an issue, in which the failure of multiple data nodes during a partial restart could cause API nodes to fail, did not properly check the validity of the associated `NdbReceiver` object before proceeding. Now in such cases an invalid object triggers handling for invalid signals, rather than a node failure. (Bug #25902137)

  References: This issue is a regression of: Bug #25092498.

- **NDB Cluster APIs:** Incorrect results, usually an empty result set, were returned when `setBound()` was used to specify a `NULL` bound. This issue appears to have been caused by a problem in gcc, limited to cases using the old version of this method (which does not employ `NdbRecord`), and is fixed by rewriting the problematic internal logic in the old implementation. (Bug #89468, Bug #27461752)

- **NDB Cluster APIs:** Released NDB API objects are kept in one or more `Ndb_free_list` structures for later reuse. Each list also keeps track of all objects seized from it, and makes sure that these are eventually released back to it. In the event that the internal function `NdbScanOperation::init()` failed, it was possible for an `NdbApiSignal` already allocated by the `NdbOperation` to be leaked. Now in such cases, `NdbScanOperation::release()` is called to release any objects allocated by the failed `NdbScanOperation` before it is returned to the free list.

  This fix also handles a similar issue with `NdbOperation::init()`, where a failed call could also leak a signal. (Bug #89249, Bug #27389894)

- **NDB Cluster APIs:** Removed the unused `TFSentinel` implementation class, which raised compiler warnings on 32-bit systems. (Bug #89005, Bug #27302881)

- **NDB Cluster APIs:** The success of thread creation by API calls was not always checked, which could lead to timeouts in some cases. (Bug #88784, Bug #27225714)

- **NDB Cluster APIs:** The file `storage/ndb/src/ndbapi/ndberror.c` was renamed to `ndberror.cpp`. (Bug #87725, Bug #26781567)

- **ndbinfo Information Database:** Counts of committed rows and committed operations per fragment used by some tables in `ndbinfo` were taken from the `DBACC` block, but due to the fact that commit signals can arrive out of order, transient counter values could be negative. This could happen if, for example, a transaction contained several interleaved insert and delete operations on the same row; in such cases, commit signals for delete operations could arrive before those for the corresponding insert operations, leading to a failure in `DBACC`.

  This issue is fixed by using the counts of committed rows which are kept in `DBTUP`, which do not have this problem. (Bug #88087, Bug #26968613)

- **NDB Client Programs:** When passed an invalid connection string, the `ndb_mgm` client did not always free up all memory used before exiting. (Bug #90179, Bug #27737906)

- **NDB Client Programs:** `ndb_show_tables` did not always free up all memory which it used. (Bug #90152, Bug #27727544)

- **NDB Client Programs:** On Unix platforms, the Auto-Installer failed to stop the cluster when `ndb_mgmd` was installed in a directory other than the default. (Bug #89624, Bug #27531186)

- **NDB Client Programs:** The Auto-Installer did not provide a mechanism for setting the `ServerPort` parameter. (Bug #89623, Bug #27539823)

- **MySQL NDB ClusterJ:** When a table containing a `BLOB` or a `TEXT` field was being queried with ClusterJ for a record that did not exist, an exception ("`The method is not valid in current blob state`") was thrown. (Bug #28536926)
- **MySQL NDB ClusterJ:** ClusterJ quit unexpectedly as there was no error handling in the `scanIndex()` function of the `ClusterTransactionImpl` class for a null returned to it internally by the `scanIndex()` method of the `ndbTransaction` class. (Bug #27297681, Bug #88989)
- Local checkpoints did not always handle `DROP TABLE` operations correctly. (Bug #27926532)

  References: This issue is a regression of: Bug #26908347, Bug #26968613.
- In some circumstances, when a transaction was aborted in the `DBTC` block, there remained links to trigger records from operation records which were not yet reference-counted, but when such an operation record was released the trigger reference count was still decremented. (Bug #27629680)
- An internal buffer being reused immediately after it had been freed could lead to an unplanned data node shutdown. (Bug #27622643)

  References: See also: Bug #28698831.
- An `NDB` online backup consists of data, which is fuzzy, and a redo and undo log. To restore to a consistent state it is necessary to ensure that the log contains all of the changes spanning the capture of the fuzzy data portion and beyond to a consistent snapshot point. This is achieved by waiting for a GCI boundary to be passed after the capture of data is complete, but before stopping change logging and recording the stop GCI in the backup's metadata.

  At restore time, the log is replayed up to the stop GCI, restoring the system to the state it had at the consistent stop GCI. A problem arose when, under load, it was possible to select a GCI boundary which occurred too early and did not span all the data captured. This could lead to inconsistencies when restoring the backup; these could be noticed as broken constraints or corrupted `BLOB` entries.

  Now the stop GCI is chosen is so that it spans the entire duration of the fuzzy data capture process, so that the backup log always contains all data within a given stop GCI. (Bug #27497461)

  References: See also: Bug #27566346.
- For `NDB` tables, when a foreign key was added or dropped as a part of a DDL statement, the foreign key metatdata for all parent tables referenced should be reloaded in the handler on all SQL nodes connected to the cluster, but this was done only on the `mysqld` on which the statement was executed. Due to this, any subsequent queries relying on foreign key metadata from the corresponding parent tables could return inconsistent results. (Bug #27439587)

  References: See also: Bug #82989, Bug #24666177.
- `ANALYZE TABLE` used excessive amounts of CPU on large, low-cardinality tables. (Bug #27438963)
- Queries using very large lists with `IN` were not handled correctly, which could lead to data node failures. (Bug #27397802)

  References: See also: Bug #28728603.
- A data node overload could in some situations lead to an unplanned shutdown of the data node, which led to all data nodes disconnecting from the management and nodes.

  This was due to a situation in which `API_FAILREQ` was not the last received signal prior to the node failure.

  As part of this fix, the transaction coordinator's handling of `SCAN_TABREQ` signals for an `ApiConnectRecord` in an incorrect state was also improved. (Bug #27381901)

  References: See also: Bug #47039, Bug #11755287.

- In a two-node cluster, when the node having the lowest ID was started using `--nostart`, API clients could not connect, failing with `Could not alloc node id at HOST port PORT_NO: No free node id found for mysqld(API)`. (Bug #27225212)

- Changing `MaxNoOfExecutionThreads` without an initial system restart led to an unplanned data node shutdown. (Bug #27169282)

  References: This issue is a regression of: Bug #26908347, Bug #26968613.

- In most cases, and especially in error conditions, `NDB` command-line programs failed on exit to free memory used by option handling, and failed to call `ndb_end()`. This is fixed by removing the internal methods `ndb_load_defaults()` and `ndb_free_defaults()` from `storage/ndb/include/util/ndb_opts.h`, and replacing these with an `Ndb_opts` class that automatically frees such resources as part of its destructor. (Bug #26930148)

  References: See also: Bug #87396, Bug #26617328.

- A query against the `INFORMATION_SCHEMA.FILES` table returned no results when it included an `ORDER BY` clause. (Bug #26877788)

- During a restart, `DBLQH` loads redo log part metadata for each redo log part it manages, from one or more redo log files. Since each file has a limited capacity for metadata, the number of files which must be consulted depends on the size of the redo log part. These files are opened, read, and closed sequentially, but the closing of one file occurs concurrently with the opening of the next.

  In cases where closing of the file was slow, it was possible for more than 4 files per redo log part to be open concurrently; since these files were opened using the `OM_WRITE_BUFFER` option, more than 4 chunks of write buffer were allocated per part in such cases. The write buffer pool is not unlimited; if all redo log parts were in a similar state, the pool was exhausted, causing the data node to shut down.

  This issue is resolved by avoiding the use of `OM_WRITE_BUFFER` during metadata reload, so that any transient opening of more than 4 redo log files per log file part no longer leads to failure of the data node. (Bug #25965370)

- Under certain conditions, data nodes restarted unnecessarily during execution of `ALTER TABLE... REORGANIZE PARTITION`. (Bug #25675481)

  References: See also: Bug #26735618, Bug #27191468.

- Race conditions sometimes occurred during asynchronous disconnection and reconnection of the transporter while other threads concurrently inserted signal data into the send buffers, leading to an unplanned shutdown of the cluster.

  As part of the work fixing this issue, the internal templating function used by the Transporter Registry when it prepares a send is refactored to use likely-or-unlikely logic to speed up execution, and to remove a number of duplicate checks for NULL. (Bug #24444908, Bug #25128512)

  References: See also: Bug #20112700.

- `ndb_restore` sometimes logged data file and log file progress values much greater than 100%. (Bug #20989106)

- Removed unneeded debug printouts from the internal function `ha_ndbcluster::copy_fk_for_offline_alter()`. (Bug #90991, Bug #28069711)

- The internal function `BitmaskImpl::setRange()` set one bit fewer than specified. (Bug #90648, Bug #27931995)

- Inserting a row into an `NDB` table having a self-referencing foreign key that referenced a unique index on the table other than the primary key failed with `ER_NO_REFERENCED_ROW_2`. This was

due to the fact that NDB checked foreign key constraints before the unique index was updated, so that the constraint check was unable to use the index for locating the row. Now, in such cases, NDB waits until all unique index values have been updated before checking foreign key constraints on the inserted row. (Bug #90644, Bug #27930382)

References: See also: Bug #91965, Bug #28486390.

- Removed all references to the C++ register storage class in the NDB Cluster sources; use of this specifier, which was deprecated in C++11 and removed in C++17, raised warnings when building with recent compilers. (Bug #90110, Bug #27705985)
- It was not possible to create an NDB table using PARTITION_BALANCE set to FOR_RA_BY_LDM_X_2, FOR_RA_BY_LDM_X_3, or FOR_RA_BY_LDM_X_4. (Bug #89811, Bug #27602352)

  References: This issue is a regression of: Bug #81759, Bug #23544301.
- Adding a [tcp] or [shm] section to the global configuration file for a cluster with multiple data nodes caused default TCP connections to be lost to the node using the single section. (Bug #89627, Bug #27532407)
- Removed a memory leak in the configuration file parser. (Bug #89392, Bug #27440614)
- Fixed a number of implicit-fallthrough warnings, warnings raised by uninitialized values, and other warnings encountered when compiling NDB with GCC 7.2.0. (Bug #89254, Bug #89255, Bug #89258, Bug #89259, Bug #89270, Bug #27390714, Bug #27390745, Bug #27390684, Bug #27390816, Bug #27396662)

  References: See also: Bug #88136, Bug #26990244.
- Node connection states were not always reported correctly by ClusterMgr immediately after reporting a disconnection. (Bug #89121, Bug #27349285)
- As a result of the reuse of code intended for send threads when performing an assist send, all of the local release send buffers were released to the global pool, which caused the intended level of the local send buffer pool to be ignored. Now send threads and assisting worker threads follow their own policies for maintaining their local buffer pools. (Bug #89119, Bug #27349118)
- When the PGMAN block seized a new Page_request record using seizeLast, its return value was not checked, which could cause access to invalid memory. (Bug #89009, Bug #27303191)
- TCROLLBACKREP signals were not handled correctly by the DBTC kernel block. (Bug #89004, Bug #27302734)
- When sending priority A signals, we now ensure that the number of pending signals is explicitly initialized. (Bug #88986, Bug #27294856)
- The internal function ha_ndbcluster::unpack_record() did not perform proper error handling. (Bug #88587, Bug #27150980)
- CHECKSUM is not supported for NDB tables, but this was not not reflected in the CHECKSUM column of the INFORMATION_SCHEMA.TABLES table, which could potentially assume a random value in such cases. Now the value of this column is always set to NULL for NDB tables, just as it is for InnoDB tables. (Bug #88552, Bug #27143813)
- Removed a memory leak detected when running ndb_mgm -e "CLUSTERLOG ...". (Bug #88517, Bug #27128846)
- When terminating, ndb_config did not release all memory which it had used. (Bug #88515, Bug #27128398)
- ndb_restore did not free memory properly before exiting. (Bug #88514, Bug #27128361)

- In certain circumstances where multiple `Ndb` objects were being used in parallel from an API node, the block number extracted from a block reference in `DBLQH` was the same as that of a `SUMA` block even though the request was coming from an API node. Due to this ambiguity, `DBLQH` mistook the request from the API node for a request from a `SUMA` block and failed. This is fixed by checking node IDs before checking block numbers. (Bug #88441, Bug #27130570)

- A join entirely within the materialized part of a semijoin was not pushed even if it could have been. In addition, `EXPLAIN` provided no information about why the join was not pushed. (Bug #88224, Bug #27022925)

  References: See also: Bug #27067538.

- All known compiler warnings raised by `-Werror` when building the `NDB` source code have been fixed. (Bug #88136, Bug #26990244)

- When the duplicate weedout algorithm was used for evaluating a semijoin, the result had missing rows. (Bug #88117, Bug #26984919)

  References: See also: Bug #87992, Bug #26926666.

- `NDB` did not compile with GCC 7. (Bug #88011, Bug #26933472)

- A table used in a loose scan could be used as a child in a pushed join query, leading to possibly incorrect results. (Bug #87992, Bug #26926666)

- When representing a materialized semijoin in the query plan, the MySQL Optimizer inserted extra `QEP_TAB` and `JOIN_TAB` objects to represent access to the materialized subquery result. The join pushdown analyzer did not properly set up its internal data structures for these, leaving them uninitialized instead. This meant that later usage of any item objects referencing the materialized semijoin accessed an initialized `tableno` column when accessing a 64-bit `tableno` bitmask, possibly referring to a point beyond its end, leading to an unplanned shutdown of the SQL node. (Bug #87971, Bug #26919289)

- In some cases, a `SCAN_FRAGCONF` signal was received after a `SCAN_FRAGREQ` with a close flag had already been sent, clearing the timer. When this occurred, the next `SCAN_FRAGREF` to arrive caused time tracking to fail. Now in such cases, a check for a cleared timer is performed prior to processing the `SCAN_FRAGREF` message. (Bug #87942, Bug #26908347)

- While deleting an element in `Dbacc`, or moving it during hash table expansion or reduction, the method used (`getLastAndRemove()`) could return a reference to a removed element on a released page, which could later be referenced from the functions calling it. This was due to a change brought about by the implementation of dynamic index memory in NDB 7.6.2; previously, the page had always belonged to a single `Dbacc` instance, so accessing it was safe. This was no longer the case following the change; a page released in `Dbacc` could be placed directly into the global page pool where any other thread could then allocate it.

  Now we make sure that newly released pages in `Dbacc` are kept within the current `Dbacc` instance and not given over directly to the global page pool. In addition, the reference to a released page has been removed; the affected internal method now returns the last element by value, rather than by reference. (Bug #87932, Bug #26906640)

  References: See also: Bug #87987, Bug #26925595.

- When creating a table with a nonexistent conflict detection function, `NDB` returned an improper error message. (Bug #87628, Bug #26730019)

- `ndb_top` failed to build with the error `"HAVE_NCURSESW_H" is not defined`. (Bug #87035, Bug #26429281)

- In a MySQL Cluster with one MySQL Server configured to write a binary log failure occurred when creating and using an `NDB` table with non-stored generated columns. The problem arose only when the product was built with debugging support. (Bug #86084, Bug #25957586)

- It was possible to create or alter a `STORAGE MEMORY` table using a nonexistent tablespace without any error resulting. Such an operation now fails with Error 3510 `ER_TABLESPACE_MISSING_WITH_NAME`, as intended. (Bug #82116, Bug #23744378)
- `ndb_restore --print-data --hex` did not print trailing 0s of `LONGVARBINARY` values. (Bug #65560, Bug #14198580)
- When the internal function `ha_ndbcluster::copy_fk_for_offline_alter()` checked dependent objects on a table from which it was supposed to drop a foreign key, it did not perform any filtering for foreign keys, making it possible for it to attempt retrieval of an index or trigger instead, leading to a spurious Error 723 (`No such table`).

Index

Symbols

--allow-pk-changes, 94, 219
--backup-password-from-stdin, 73, 199
--config-binary-file, 33, 163
--config-file, 61, 188
--configdir, 61, 188
--decrypt, 80, 206
--decrypt-password-from-stdin, 73, 199
--diff-default, 33, 124, 163, 247
--disable-indexes, 61, 104, 188, 229
--encrypt-backup, 73, 199
--encrypt-password-from-stdin, 73, 199
--help, 104, 229
--ignore-extended-pk-changes, 94, 219
--include-stored-grants, 104, 229
--initial, 124, 247
--ndb-index-stat-enable, 61, 188
--ndb-log-fail-terminate, 94, 219
--ndb-nodegroup-map, 61, 188
--ndb-schema-dist-timeout, 119, 243
--nostart, 136, 259
--rebuild-indexes, 104, 229
--remap-column, 94, 219
--restore-epoch, 119, 243
--upgrade=force, 104, 229
.ctl files, 112, 236
.deb, 119, 243
.frm, 124, 247
.frm files, 99, 224

A

aborted transactions, 136, 259
AbortOption, 99, 224
ABORT_BACKUP_ORD, 119, 243
AbstractQueryPlan, 49, 177
adaptive spin, 27, 158
ADD DATAFILE, 124, 247
add nodes, 57, 80, 185, 206
AES_ENCRYPT(), 10
ALGORITHM=COPY, 104, 229
AllowUnresolvedHostNames, 86, 212
ALTER, 104, 229
ALTER LOGFILE GROUP, 124, 247

ALTER TABLE, 21, 42, 49, 57, 61, 67, 80, 86, 94, 104, 112, 153, 172, 177, 185, 188, 194, 206, 212, 219, 229, 236
ALTER TABLE INPLACE, 86, 212
ALTER TABLESPACE, 104, 119, 124, 229, 243, 247
ANALYZE TABLE, 124, 247
antijoins, 94, 219
ANTLR, 99, 224
API nodes, 17, 24, 150, 155
API timeouts, 7
ApiConnectRecord, 7
ApiFailureHandlingTimeout, 4
Apple Silicon support, 42, 172
Arbitration, 104, 229
ARM, 112, 236
arrays, 49, 177
auto-synchronization, 80, 94, 99, 206, 219, 224
autoincrement, 124, 247
automatic synchronization, 104, 229
AutomaticThreadConfig, 12, 49, 57, 80, 145, 177, 185, 206

B

backports, 17, 150
backup, 4, 67, 73, 94, 104, 119, 124, 136, 194, 199, 219, 229, 247, 259
BACKUP, 112, 236
Backup block, 112, 236
backup ID, 73, 199
BackupLogBufferSize, 104, 229
backups, 33, 124, 163, 247
batched_key_access, 57, 185
batching, 86
BINARY, 12, 119, 145
binary log, 12, 57, 67, 136, 185, 194, 259
binary log injector, 17, 67, 150, 194
binary log rotation, 7
binary logging, 17, 86, 150
bind address, 17, 150
Binlog_cache_disk_use, 33, 163
binlog_cache_size, 17, 150
BIT, 17, 61, 188
bitmaps, 67, 194
BitmaskImpl::setRange(), 136, 259
BLOB, 67, 86, 119, 124, 133, 136, 194, 212, 247, 256, 259
Blob, 86
blob operations, 49, 177
blobs, 49, 67, 86, 177, 194, 212
BLOB_INLINE_SIZE, 7, 10, 42, 172
block thread scheduler, 49, 177
blocking, 24, 155
BuildIndexThreads, 136, 259
bus errors, 49, 177

C

case sensitivity, 94, 219
certifi, 15
changes
  NDB Cluster, 145
character sets, 112, 236

checkpoints, 24, 136, 155, 259
CHECKSUM, 136, 259
Clang, 21, 80, 153, 206
clang, 124, 247
ClassicFragmentation, 80, 206
Cleaner, 4
CLOCK_MONOTONIC, 80, 206
close(), 124, 247
closing tables, 73, 199
ClusterTransactionImpl, 136, 259
CMake3, 124, 247
CMVMI, 119, 243
CM_REGREF, 61, 188
collations, 136, 259
COLUMN_FORMAT, 119, 243
comparison methods, 112, 236
comparisons, 24, 155
compilation, 112, 236
compile-cluster, 4
compiling, 12, 17, 21, 37, 42, 49, 61, 67, 73, 80, 86, 94, 112, 119, 124, 136, 145, 150, 153, 167, 172, 177, 188, 194, 199, 206, 212, 219, 236, 243, 247, 259
CompressedLCP, 49, 177
compression, 33, 163
computeXorChecksum(), 49, 177
concurrent operations, 124, 247
condition pushdown, 12, 37, 61, 67, 73, 99, 112, 124, 145, 167, 188, 194, 199, 224, 236, 247
config.ini, 49, 136, 177, 259
configuration, 10, 73, 104, 112, 124, 199, 229, 236, 247
configuration handling, 112, 236
conflict resolution, 21, 153
connection timeouts, 21, 153
connections, 24, 155
CONTINUEB signal, 57, 185
CopyFrag, 12, 33, 145, 163
CopyGCI, 4
COPY_FRAGREQ, 42, 172
correlation IDs, 124, 247
CPU spin, 73, 199
cpustat, 124, 247
CREATE INDEX, 61, 188
CREATE NODEGROUP, 67, 194
CREATE TABLE, 49, 61, 67, 80, 112, 119, 124, 133, 136, 177, 188, 194, 206, 236, 247, 256, 259
CSV, 42, 94, 172
curl, 73

D

data dictionary, 24, 27, 42, 104, 119, 124, 133, 136, 155, 158, 172, 229, 243, 247, 256, 259
data files, 99, 112, 224, 236
data node, 133, 256
data node shutdown, 136, 259
data nodes, 21, 27, 33, 37, 42, 104, 112, 136, 153, 158, 163, 167, 172, 229, 236, 259
data pages, 112, 236
database object names, 73, 199

DataMemory, 33, 112, 163, 236
Date, 99, 224
DATETIME, 124, 247
DBACC, 136, 259
Dbacc::getLastAndRemove(), 136, 259
Dbacc::startNext(), 133, 256
DBDICT, 27, 57, 158, 185
DBDIH, 73, 80, 199, 206
DBLQH, 12, 80, 86, 145, 206, 212
Dblqh::sendKeyinfo20(), 136, 259
DBSPJ, 7, 17, 27, 86, 99, 119, 124, 150, 158, 212, 224, 243, 247
DBTC, 7, 12, 17, 24, 42, 119, 124, 136, 145, 150, 155, 172, 243, 247, 259
Dbtc::initData(), 104, 229
DBTUP, 37, 104, 133, 136, 167, 229, 256, 259
DbtupBuffer.cpp, 104, 229
DbUtil, 67, 194
DDL, 80, 86, 119, 124, 206, 212, 243, 247
DedicatedNode, 104, 229
DELETE, 67, 194
demos, 124, 247
deprecation, 61, 188
detailed-info, 37, 167
Dictionary::getEvent(), 42, 172
Dictionary::releaseEvent(), 42, 172
dictionary_columns, 49, 177
dictionary_tables, 49, 177
dict_obj_tree, 73, 199
directio, 73, 199
disable-indexes, 15, 148
disconnection, 124, 247
disk format, 104, 229
DiskDataUsingSameDisk, 104, 229
diskstat, 104, 229
diskstats_1sec, 104, 229
DISK_RECORDS, 27, 158
distributed privileges, 112, 124, 236, 247
distribution, 136, 259
DIVERIFYREQ, 57, 185
DNS cache, 73, 199
Docker, 67, 194
dojo, 86, 94, 212, 219
DomainTypeHandler, 80, 206
DROP DATABASE, 124
DROP NODEGROUP, 33, 163
DROP TABLE, 86, 119, 124, 136, 212, 243, 247, 259
DROP_TRIG_IMPL_REQ, 133, 256
DUMP, 104, 112, 229, 236
DUMP 11001, 57, 185
DUMP 9988, 104, 229
DUMP 9989, 104, 229
DUMP commands, 21, 153
DumpPageMemoryOnFail, 94, 219
duplicate weedout, 136, 259
dynamic index memory, 136, 259

E

element deletion, 136, 259
EnableRedoControl, 86, 212

encrypted backup, 73, 199
EncryptedFilesystem, 37, 167
EncryptedFileSystem, 42, 172
encryption, 33, 37, 163, 167
endianness, 124, 247
ENOMEM, 17, 150
epochs, 119, 243
eq_ref, 119, 124, 243, 247
error 1297, 57, 185
error codes, 133, 256
error handling, 15, 42, 67, 112, 148, 172, 194, 236
error log, 67, 73, 194, 199
error messages, 104, 112, 229, 236
error reporting, 61, 188
errors, 17, 21, 24, 37, 73, 104, 112, 119, 133, 136, 150, 153, 155, 167, 199, 229, 236, 243, 256, 259
ERROR_FILE_NOT_FOUND, 49, 177
ERROR_PATH_NOT_FOUND, 49, 177
ER_NO_REFERENCED_ROW_2, 136, 259
ER_TABLESPACE_MISSING_WITH_NAME, 136, 259
ER_TRANSACTION_FAILED, 67, 194
Event, 27, 158
event buffer, 17, 67, 150, 194
event data buffer, 49, 177
event subscriptions, 61, 188
EventBuffer, 15, 148
EventBytesRecvdCount, 49, 177
events, 17, 49, 150, 177
examples, 57, 73, 112, 185, 199, 236
examples (NDB API), 136, 259
execSCAN_FRAGREQ(), 49, 177
exit(), 119, 243
expat, 15
EXPLAIN, 37, 86, 136, 167, 212, 259

F

failure handling, 10
files, 49, 57, 177, 185
FILES, 136, 259
FILES table, 104, 229
filesort, 99, 224
FIXED, 119, 243
foreign key constraint, 124, 247
foreign keys, 27, 99, 104, 112, 133, 136, 158, 224, 229, 236, 256, 259
foreign_keys, 49, 177
foreign_key_checks, 80, 206
free memory, 15, 148
frm files, 136, 259
FULLY REPLICATED, 86, 212

G

gcc, 136, 259
GCC 8, 133, 256
GCI, 17, 94, 133, 150, 219, 256
GCI boundary, 136, 259
GCP, 21, 73, 104, 153, 199, 229
GCP stall, 33, 163

GCP stalls, 21, 153
GCP_SAVEREQ, 86, 212
generated columns, 112, 236
getBlobHashKey(), 49, 177
getCreateSchemaVersion(), 112, 236
getExtraMetadata(), 136, 259
getGCIEventOperations(), 27, 158
GET_TABLEID_REQ, 61, 188
global checkpoints, 27, 80, 158, 206
global schema lock, 12, 17, 104, 112, 124, 145, 150, 229, 236, 247
grants, 17
GSN_STOP_REQ, 86, 212
GSN_TRANSID_AI, 42, 172

H

hash scan, 17
hashindexes, 80, 206
hashing, 136, 259
hash_maps, 49, 177
HA_ERR_TX_FAILED, 67, 194
ha_ndbcluster, 119, 243
ha_ndbcluster::copy_fk_for_offline_alter(), 136, 259
ha_ndbcluster::unpack_record(), 136, 259
heartbeats, 15, 148
help text, 15, 148
host names, 112, 236
HostName, 57, 185

I

ID allocation, 104, 229
identifiers, 112, 236
idxbld, 136, 259
Important Change, 15, 24, 37, 42, 49, 57, 61, 80, 86, 94, 99, 104, 112, 119, 124, 136, 148, 155, 167, 172, 177, 185, 188, 206, 212, 219, 224, 229, 236, 243, 247, 259
IN, 136, 259
IN(), 24, 155
include-stored-grants, 4
INCL_NODECONF, 104, 229
Incompatible Change, 104, 124, 229, 247
index length, 124, 247
index statistics, 10, 27, 33, 37, 42, 57, 61, 67, 73, 99, 112, 158, 163, 167, 172, 185, 188, 194, 199, 224, 236
indexes, 49, 94, 177, 219
index_columns, 49, 177
INFORMATION_SCHEMA, 136, 259
INFORMATION_SCHEMA.FILES, 124, 136, 247, 259
INFORMATION_SCHEMA.TABLES, 124, 247
InitFragmentLogFiles, 136, 259
initial start, 94, 219
InitialNoOfOpenFiles, 124, 247
INPLACE, 12, 67, 86, 104, 119, 145, 194, 212, 229, 243
INSERT IGNORE, 67, 194
insert operations, 133, 256
InstallDirectory, 136, 259
integer, 49, 177
invisible indexes, 104, 229

InvocationHandler, 4
IPv6, 61, 67, 86, 188, 194, 212
IPv6 support, 24, 155
isnan(), 99, 224

J

jam(), 80, 206
Java 11 deprecation warnings, 99, 224
Java versions, 99, 224
job buffers, 37, 80, 167, 206
job scheduler, 37, 167
join pushdown, 73, 199
joins, 24, 37, 99, 112, 155, 167, 224, 236
JOIN_TAB, 136, 259

K

KeepAliveSendInterval, 61, 188
kill -9, 80, 206

L

LateAlloc, 94, 219
LCP, 73, 86, 99, 104, 112, 124, 199, 212, 224, 229, 236, 247
LCP pause, 133, 256
LCPs, 80, 206
LCP_COMPLETE_REP, 104, 124, 229, 247
LCP_SCANNED_BIT, 33, 163
LCP_STATUS_IDLE, 124, 247
LCP_TAB_SAVED, 112, 236
LDM, 104, 229
LEFT JOIN, 27, 158
less than, 124, 247
libndbclient-devel, 133, 256
libndbclient.so, 133, 256
libssh, 12
libssl.so, 133, 256
libxml2, 12
LIMIT, 61, 119, 188, 243
Linux, 73, 199
List::clear(), 49, 177
listEvents(), 49, 177
listObjects(), 73, 199
lld, 12, 145
lock contention, 124, 247
locking, 104, 112, 229, 236
locks, 67, 80, 104, 119, 124, 194, 206, 229, 243, 247
log buffer, 136, 259
log file groups, 133, 256
log sequence number, 80, 206
logbuffers, 136, 259
LOGFILE GROUP, 124, 247
logging, 7, 12, 17, 24, 42, 49, 57, 67, 73, 80, 99, 104, 145, 150, 155,
172, 177, 185, 194, 199, 206, 224, 229
logs, 15, 148
log_replica_updates, 17, 150
long signals, 104, 119, 229, 243
LongMessageBuffer, 67, 194
LONGVARBINARY, 136, 259

LooseScan, 136, 259
lower_case_table_names, 99, 104, 112, 224, 229, 236
LQHKEYREQ, 86, 112, 212, 236
LQH_TRANSCONF, 4

M

macOS, 7, 42, 80, 104, 112, 172, 206, 229, 236
master node, 104, 229
MASTER_LCPCONF, 119, 243
MASTER_LCP_REQ, 119, 243
materialized semijoin, 136, 259
MaxAllocate, 61, 188
MaxBufferedEpochs, 124, 247
MaxDiskDataLatency, 104, 229
MaxDiskWriteSpeedOwnRestart, 86, 212
MaxDMLOperationsPerTransaction, 27, 119, 158, 243
MaxNoOfConcurrentOperations, 49, 112, 119, 177, 236, 243
MaxNoOfExecutionThreads, 124, 136, 247, 259
MaxNoOfOpenFiles, 124, 247
MaxNoOfOrderedIndexes, 124, 247
MaxNoOfSubscriptions, 67
MaxNoOfTables, 124, 247
MaxNoOfUniqueHashIndexes, 124, 247
maxRecordSize, 21, 153
MAX_BLOB_PART_SIZE, 42, 172
MAX_EXECUTION_TIME, 119, 243
MAX_NODES, 112, 236
MAX_REPLICAS, 80, 206
MAX_ROWS, 104, 229
memcache, 119, 243
meminfo, 73, 199
memory allocation, 104, 229
memory usage, 94, 124, 219, 247
metadata, 80, 112, 124, 133, 136, 206, 236, 247, 256, 259
metadata lock, 17
metadata locks, 104, 229
metadata synchronization, 112, 236
mgmd, 73, 199
Microsoft Windows, 27, 49, 57, 61, 73, 80, 104, 136, 158, 177, 188, 199, 206, 229, 259
monitor process, 136, 259
multi-byte, 119, 243
my.cnf, 61, 188
MySQL Enterprise Monitor, 57, 185
MySQL NDB ClusterJ, 4, 10, 27, 33, 42, 80, 99, 112, 124, 133, 136, 158, 163, 172, 206, 224, 236, 247, 256, 259
mysql.ndb_apply_status table, 12
mysqlbinlog, 67, 194
mysqld, 15, 42, 67, 119, 124, 148, 172, 194, 243, 247
mysqldump, 124, 247

N

NDB Client Programs, 4, 10, 12, 15, 17, 21, 33, 37, 80, 86, 99, 112, 136, 145, 148, 150, 153, 163, 167, 206, 212, 224, 236, 259
NDB Cluster, 4, 7, 10, 12, 15, 17, 21, 24, 27, 33, 37, 42, 49, 57, 61, 67, 73, 80, 86, 94, 99, 104, 112, 119, 124, 133, 136, 145, 148, 150,

153, 155, 158, 163, 167, 172, 177, 185, 188, 194, 199, 206, 212, 219, 224, 229, 236, 243, 247, 256, 259
NDB Cluster APIs, 7, 10, 17, 21, 24, 27, 42, 49, 57, 61, 67, 73, 80, 86, 112, 119, 124, 136, 150, 153, 155, 158, 172, 177, 185, 188, 194, 199, 206, 212, 236, 243, 247, 259
NDB Disk Data, 7, 42, 86, 94, 99, 104, 112, 119, 124, 133, 136, 172, 212, 219, 224, 229, 236, 243, 247, 256, 259
NDB Operator, 17, 150
NDB programs, 136, 259
NDB Replication, 4, 7, 12, 17, 21, 27, 42, 49, 57, 67, 80, 86, 119, 124, 136, 172, 259
NDB$BLOB, 124
NDB$MAX_DEL_WIN_INS(), 42
NDB$MAX_INS(), 42
ndb-applier-allow-skip-epoch, 57
ndb-batch-size, 49, 177
ndb-common, 67, 194
ndb-log-transaction-dependency, 27
ndb-wait-connected, 61, 188
ndb-wait-setup, 15, 61, 148, 188
ndb.apply_status table, 27
Ndb::pollEvents2(), 21, 153
ndbapi_array_simple, 136, 259
ndbapi_array_using_adapter, 136, 259
ndbapi_simple_dual, 136, 259
ndbcluster_print_error(), 124, 247
ndbd, 12, 49, 57, 73, 145, 177, 185, 199
NdbDictionary, 61, 188
NdbDictionaryImpl.cpp, 12, 145
ndbd_exit(), 42, 172
ndberr, 133, 256
NdbEventBuffer, 49, 177
NdbEventOperation, 24, 155
NDBFS, 21, 42, 172
NdbfsDumpRequests, 124, 247
ndbimport, 94
NdbIndexScanOperation::setBound(), 136, 259
ndbinfo, 57, 80, 99, 185, 206, 224
ndbinfo Information Database, 4, 7, 12, 15, 17, 37, 73, 136, 145, 148, 150, 167, 199, 259
ndbinfo.cluster_locks, 67, 194
ndbinfo.index_stats, 57, 185
ndbinfo.restart_info, 17, 150
ndbinfo.threads, 80, 206
NdbInterpretedCode, 24, 42, 61, 112, 155, 172, 188, 236
ndbmemcache, 80, 86, 133, 206, 212, 256
ndbmtd, 49, 80, 86, 124, 133, 177, 206, 212, 247, 256
NdbOperation, 24, 155
ndbout, 133, 256
NdbReceiver, 136, 259
NdbReceiver::unpackNdbRecord(), 80, 206
NdbReceiverBuffer, 99, 224
NdbRecord, 10
NdbScanFilter, 61, 112, 133, 188, 236, 256
NdbScanFilter::setSqlCmpSemantics(), 67, 194
NdbScanOperation, 136, 259
NdbSpin, 27, 158
NDBT, 112, 119, 236, 243

NdbThread_SetThreadPrio, 42, 172
NdbTransaction, 86, 124, 212, 247
ndbxfrm, 33, 37, 49, 67, 163, 167, 177, 194
Ndb_api_read_row_count, 112, 236
Ndb_api_wait_nanos_count, 42, 172
ndb_apply_status, 27, 57, 119, 158
ndb_autoincrement_prefetch_sz, 104, 229
ndb_binlog_index, 124, 247
ndb_blob_tool, 37, 57, 99, 167, 185, 224
Ndb_cluster_connection, 7, 24, 155
NDB_COLUMN, 42, 172
ndb_config, 7, 33, 73, 104, 124, 136, 163, 199, 229, 247, 259
Ndb_config_generation, 73, 199
ndb_delete_all, 112, 236
ndb_desc, 73, 94, 199, 219
ndb_drop_table, 104, 119, 229, 243
ndb_import, 27, 42, 49, 57, 67, 94, 112, 119, 158, 172, 177, 185, 194, 219, 236, 243
Ndb_index_stat, 49, 177
ndb_join_pushdown, 33, 163
NDB_LE_EventBufferStatus, 67, 194
ndb_logevent_get_next2(), 119, 243
Ndb_logevent_type, 67, 194
ndb_log_apply_status, 12
ndb_log_bin, 124, 247
ndb_log_cache_size, 10
ndb_log_transaction_compression, 37, 167
ndb_log_transaction_compression_level_zstd, 37, 167
ndb_log_update_as_write, 21
ndb_log_update_minimal, 21
Ndb_metadata_blacklist_size, 86, 94, 112, 212, 219, 236
Ndb_metadata_change_monitor, 124, 247
ndb_metadata_check, 4, 104, 124, 229, 247
ndb_metadata_check_interval, 24, 124, 155, 247
Ndb_metadata_detected_count, 124, 247
Ndb_metadata_excluded_count, 86, 212
ndb_metadata_sync, 86, 99, 104, 212, 224, 229
Ndb_metadata_synced_count, 112, 236
ndb_mgm, 119, 136, 243, 259
ndb_mgmd, 15, 27, 33, 57, 61, 67, 73, 86, 112, 119, 133, 136, 148, 158, 163, 185, 188, 194, 199, 212, 236, 243, 256, 259
ndb_mgm_create_logevent_handle(), 61, 188
NDB_MGM_NODE_TYPE_UNKNOWN, 61, 188
ndb_milli_sleep(), 112, 236
ndb_my_error(), 133, 256
Ndb_opts, 136, 259
ndb_print_backup_file, 17, 67, 86, 150, 194, 212
ndb_print_sys_file, 80, 206
ndb_read_backup, 104, 229
ndb_redo_log_reader, 15, 148
Ndb_ReloadHWInfo(), 73, 199
ndb_replica_batch_size, 42
ndb_replica_blob_write_batch_bytes, 42
ndb_restore, 12, 21, 24, 27, 37, 42, 49, 61, 67, 73, 80, 86, 94, 99, 104, 112, 119, 124, 133, 136, 145, 155, 158, 167, 172, 177, 188, 194, 199, 206, 212, 219, 224, 229, 236, 243, 247, 256, 259
ndb_row_checksum, 133, 256
ndb_schema, 27, 99, 104, 158, 224, 229

ndb_schema_dist_lock_wait_timeout, 112, 236
ndb_schema_object, 4
NDB_SCHEMA_OBJECT, 12, 119, 145, 243
Ndb_schema_participant_count, 4
ndb_schema_share, 119, 243
ndb_secretsfile_reader, 37, 167
ndb_select_all, 21, 153
ndb_setup.py, 80, 86, 119, 206, 212, 243
NDB_SHARE, 112, 236
ndb_show_tables, 99, 136, 224, 259
ndb_size.pl, 10
ndb_sql_metadata, 49, 177
NDB_STORED_USER, 17, 33, 57, 61, 67, 73, 86, 104, 112, 163, 185, 188, 199, 212, 229, 236
ndb_sync_excluded_objects, 94, 219
ndb_sync_pending_objects, 94, 219
NDB_TABLE, 94, 219
ndb_top, 33, 136, 163, 259
Ndb_trans_hint_count_session, 119, 243
ndb_waiter, 15, 27, 99, 148, 158, 224
ndb_wait_setup, 104, 229
NDB_WIN32, 67, 194
ndinfo.index_stats, 57, 185
neighbor nodes, 15, 148
newtonapi, 124, 247
NO PAD collations, 124, 247
node failure handling, 49, 73, 177, 199
node ID allocation, 73, 112, 199, 236
node IDs, 7, 49, 112, 177, 236
node logs, 61, 188
node recovery, 80, 206
node restarts, 57, 86, 94, 185, 212, 219
node shutdown, 10, 33, 73, 163, 199
node takeover, 112, 236
Node.JS, 27, 158
Node.js, 42, 61, 67, 73, 86, 172, 188, 194, 199, 212
NodeGroup, 67, 86, 194, 212
NodeGroupTransporters, 99, 224
NODE_FAILREP, 104, 229
NoOfReplicas, 67, 86, 104, 194, 212, 229
NOT BETWEEN, 124, 247
NOT IN, 124, 247
NOWAIT, 21, 153
NO_OF_BUCKETS, 61
NULL, 49, 124, 133, 136, 177, 247, 256, 259
NULL comparison, 67, 194
num-slices, 99, 224
NumCPUs, 12, 80, 145, 206

O

ODirect, 12, 136, 145, 259
ODirectSyncFlag, 136, 259
OM_WRITE_BUFFER, 136, 259
ON DELETE CASCADE, 124, 247
online alter, 27, 158
online DDL, 112, 236
online operations, 15, 148
online table reorganization, 17, 150

online upgrades, 136, 259
OOBR, 4
OO_NOWAIT, 112, 236
OpenJPA, 10
optimizer, 119, 243
option handling, 136, 259
ORDER BY, 112, 136, 236, 259
OS X, 99, 224
out of buffer errors, 4
O_DIRECT, 73, 86, 104, 199, 212, 229

P

Packaging, 15, 33, 67, 73, 86, 99, 119, 124, 133, 163, 194, 199, 212, 224, 243, 247, 256
packaging, 17, 57, 94, 219
packed send, 80, 206
parallelism, 27, 99, 124, 158, 224, 247
partial LCP, 80, 104, 206, 229
partial restarts, 104, 229
PartionCount, 73, 199
Partitioning, 112, 236
PartitionsPerNode, 80, 206
PARTITION_BALANCE, 136, 259
Performance, 49, 104, 133, 177, 229, 256
performance, 119, 243
Performance Schema, 42, 172
PGMAN, 80, 86, 104, 136, 206, 212, 229, 259
pgman_time_track_stats, 104, 229
PK reads, 86, 212
plugin threads, 49, 177
PortNumber, 73, 199
PreferIPVersion, 67, 194
primary key updates, 21
primary keys, 21, 24, 155
privileges, 42, 172
processes, 27, 158
processes table, 136, 259
PURGE BINARY LOGS, 4
pushdown, 119, 124, 243, 247
pushdown joins, 94, 219
pushed conditions, 124, 247
pushed joins, 49, 61, 112, 124, 177, 188, 236, 247

Q

QEP_TAB, 136, 259
QMGR, 4, 17, 99, 150, 224
query memory, 124, 247

R

race, 136, 259
race condition, 124, 247
race conditions, 119, 243
range checks, 57, 185
range scans, 57, 185
readln_socket(), 17, 150
READ_BACKUP, 37, 104, 167, 229
read_cost(), 57, 185

read_only, 17
realtime break, 27, 158
RealtimeScheduler, 86, 212
receive thread, 49, 80, 177, 206
redo log, 42, 67, 73, 124, 136, 172, 194, 199, 247, 259
redo log part metadata, 136, 259
redo logs, 49, 177
RedoOverCommitCounter, 104, 229
RedoOverCommitLimit, 104, 229
ref, 119, 243
REGCONF, 24, 155
register, 136, 259
REGREQ, 24, 155
releaseGlobal(), 17, 150
REORGANIZE PARTITION, 33, 136, 163, 259
REORG_MOVED, 17, 150
replica_allow_batching, 42, 172
replica_parallel_workers, 27
REPORT BACKUPSTATUS, 4
RESET MASTER, 136, 259
RESET REPLICA, 80
RESET SLAVE, 80
resource allocation, 112, 124, 236, 247
resource usage, 136, 259
RESTART, 124, 247
restarts, 12, 61, 73, 80, 112, 136, 145, 188, 199, 206, 236, 259
restore, 49, 177
rollback, 86, 212
rolling restart, 17, 150
row buffers, 86, 212
row ID, 124, 247
ROW_FORMAT, 136, 259
RPMs, 67, 194

S

SafeCounter, 124, 247
SavedEvent, 119, 243
scanIndex(), 136, 259
scans, 7, 119, 133, 243, 256
SCANTABREQ, 94, 219
SCAN_FRAGCONF, 136, 259
SCAN_FRAGREF, 136, 259
SCAN_FRAGREQ, 133, 136, 256, 259
schema distribution, 17, 57, 73, 86, 112, 119, 150, 185, 199, 212, 236, 243
schema operations, 27, 61, 158, 188
schema synchronization, 104, 133, 229, 256
schema UUID, 104, 229
SCHEMA_UUID_VALUE_LENGTH, 99, 224
scratch buffer, 27, 158
SDI, 94, 219
SELECT, 104, 124, 229, 247
semijoin, 136, 259
send buffer, 136, 259
send buffers, 136, 259
send threads, 17, 150
SendBuffer, 15, 148
ServerPort, 136, 259
setExtraMetadata(), 136, 259

SET_LOGLEVELORD, 57, 185
shared users, 99, 224
SharedGlobalMemory, 104, 119, 229, 243
SHM, 10, 136, 259
ShmSize, 104, 229
SHOW, 119, 243
signal data, 61, 188
signal dump, 4
signal fragments, 33, 163
signal memory, 73, 199
signals, 33, 42, 136, 163, 172, 259
SignalSender, 104, 229
SIGTERM, 27, 73, 158, 199
SimulatedBlock, 33, 163
singal IDs, 27, 158
single user mode, 61, 188
SINGLE USER MODE, 119, 243
SingleUserMode, 119, 243
slave-skip-errors, 57
slice-id, 99, 224
SNAPSHOTEND, 119, 243
SNAPSHOTSTART, 119, 243
sockets, 17, 150
Solaris, 33, 73, 80, 104, 163, 199, 206, 229
Solaris/x86, 112, 236
source code, 17, 150
SpinMethod, 73, 99, 199, 224
SPJ, 33, 49, 57, 80, 99, 136, 163, 177, 185, 206, 224, 259
SQL node, 124, 247
SQL nodes, 49, 67, 177, 194
SSL, 15, 148
ssl_write(), 17, 150
START_LCP_REQ, 104, 229
static_assert(), 49, 177
statistics, 49, 73, 177, 199
std::max(), 67, 194
std::min(), 67, 194
STMT_END_F, 7
STOP, 15, 124, 148, 247
stop GCI, 136, 259
STORAGE MEMORY, 136, 259
stored grants, 49, 67, 177, 194
store_table(), 27, 158
STRAIGHT_JOIN, 119, 243
strlen(), 80, 206
subscription logs, 119, 243
SUB_GCP_COMPLETE_REP, 124, 247
SUB_STOP_REQ, 133, 256
SUMA, 33, 37, 61, 86, 133, 136, 163, 167, 188, 212, 256, 259
swap, 104, 229
synchronization, 73, 112, 119, 199, 236, 243
sysfile, 86, 212
sysfiles, 112, 236
SYSTAB_0, 124, 247
system restarts, 7
system tables, 57, 185

T

table creation, 73, 199
table discovery, 112, 124, 236, 247
table reorganization, 57, 112, 185, 236
table statistics, 57, 185
Table::getColumn(), 86, 212
Table::getExtraMetadata(), 104, 229
tableno, 136, 259
TABLESPACE, 94, 219
tablespaces, 112, 119, 133, 236, 243, 256
TABLE_READ_ONLY, 112, 236
TAB_SCANREF, 37, 167
takeover, 119, 243
TC, 21, 153
TCKEYREQ, 119, 243
TcpSpinTime, 99, 224
TCRELEASEREQ, 7
TCROLLBACKREP, 136, 259
temporal data types, 124, 247
temporary errors, 119, 243
testing, 133, 256
testsuite, 4
TEXT, 124, 133, 136, 247, 256, 259
TE_ALTER, 124, 247
TFSentinel, 136, 259
thread creation, 136, 259
thread priority, 49, 177
ThreadConfig, 42, 49, 80, 124, 136, 172, 177, 206, 247, 259
timeouts, 86, 212
TIMESTAMP, 119, 243
timestamps, 67, 194
TINYBLOB, 119, 243
TLS, 21, 153
trace logs, 27, 158
transaction coordinator, 12, 112, 145, 236
transaction memory, 42, 172
TransactionMemory, 104, 229
transactions, 15, 27, 49, 104, 112, 124, 148, 158, 177, 229, 236, 247
TRANSID_AI, 86, 212
TransientPool, 49, 177
TransporterRegistry, 21, 153
TransporterRegistry::prepareSendTemplate(), 136, 259
transporters, 15, 17, 27, 148, 150, 158
transporter_details, 12, 15, 145, 148
triggers (NDB), 133, 256
TRIX, 10
TRPMAN, 15, 148
truncation, 124, 247
tuple corruption, 133, 256
TUX scans, 61, 188
TwoPassInitialNodeRestartCopy, 133, 136, 256, 259
type conversion, 119, 243
TYPE_NOTE_TIME_TRUNCATED, 33, 163
TYPE_NOTE_TRUNCATED, 33, 163

U

Ubuntu, 17, 99, 150, 224

UCA-9.0, 124, 247
undo files, 124, 247
undo log, 80, 86, 206, 212
undo log file groups, 133, 256
UndoDataBuffer, 61, 188
UndoIndexBuffer, 61, 188
UNIQUE, 57, 185
UPDATE, 61, 188
updates, 99, 224
upgrade, 86, 212
upgrades, 17, 37, 42, 57, 73, 86, 94, 104, 112, 119, 124, 133, 150, 167, 172, 185, 199, 212, 219, 229, 236, 243, 247, 256
upgrades and downgrades, 112, 236
using filesort, 119, 243
USING HASH, 57, 185
UTF8, 27, 158

V

VARCHAR, 37, 167
version_comment, 133, 256
VIRTUAL, 119
V_QUERY, 24, 155

W

wait_until_ready(), 124, 247
warnings, 15, 24, 136, 148, 155, 259
watchdog, 4
Windows, 112, 236
WiX, 104, 229
WRITESET, 27