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 NDB Cluster. 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: 2021-10-22 (revision: 23568)

Table of Contents

Preface and Legal Notices

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

Legal Notices

Copyright © 1997, 2021, Oracle and/or its affiliates.

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.

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.

If this is software 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" or "commercial computer software documentation" 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.

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.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit
https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

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 NDB Cluster.

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.

- 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)

- 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.

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 Distributed MySQL Privileges with 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)

- 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 #27150740)

- 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 NDB Cluster.

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*.

- `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.

  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 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)

- 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.

  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 NDB Cluster.

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 NDB Cluster.

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.

- 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.

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 NDB Cluster.

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`
`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.

- 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.

- This release adds the possibility of configuring the threads for multi-threaded 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.

- 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.

- 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).

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)

- 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 NDB Cluster.

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

- 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.

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)

- 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.

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 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 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.

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 NDB Cluster.

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 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)

- 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)

- `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)

- 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.

- 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.

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_TABLE 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 NDB Cluster.

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)

- **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.

- **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.

- **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.

- 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.

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`.

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)

- 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 NDB Cluster.

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)

- **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)

- **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.

- **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_TABLE Options, as well as the description of the `ndb_read_backup` system variable.

- **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.

- 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.

- 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.

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 NDB Cluster.

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`. Distributed MySQL Privileges with 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.

  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.

  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.

- **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).

- **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.

- **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.

- `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)

- 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.

- `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.

  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.

- 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.

- Columns of `NDB` tables can now be renamed online, using `ALGORITHM=INPLACE`.

  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.

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 NDB Cluster.

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.

- **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:** 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.

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 NDB Cluster.

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](#).

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)

  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)

- The `compile-cluster` script included in the `NDB` source distribution no longer supports in-source builds.

- Building with `CMake3` is now supported by the `compile-cluster` script included in the `NDB` source distribution.

- 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`.

- 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.

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 NDB Cluster.

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 NDB Cluster.

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.

- **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.

- **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.

- 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)

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 NDB Cluster.

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)

- **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`.

- **NDB Cluster APIs:** Added the `Table` methods `getExtraMetadata()` and `setExtraMetadata()`.

- `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).

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 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 8.0.26 (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*.

- `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.

  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 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)

- 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.

  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)

- 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.

- 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.

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)

- 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`
`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`

Deprecated variable	Replacement
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.

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.

- 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.

- This release adds the possibility of configuring the threads for multi-threaded 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.

- 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.

- 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).

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)

- 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)

- 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.

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)

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.

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 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 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.

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 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)

- 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)

- `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)

- 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.

- 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.

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_TABLE 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)

- **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.

- `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.

- 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.

  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`.

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)

- 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)

- 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)

- **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)

- **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.

- **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_TABLE Options, as well as the description of the `ndb_read_backup` system variable.

- **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.

- 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.

- 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.

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`. Distributed MySQL Privileges with 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.

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.

  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.

- **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).

- **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.

- **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.

- `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)

- 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.

- `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.

  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.

- 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.

- Columns of `NDB` tables can now be renamed online, using `ALGORITHM=INPLACE`.

  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)

- 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.

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.

- **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.

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](#).

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)

  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)

- The `compile-cluster` script included in the `NDB` source distribution no longer supports in-source builds.

- Building with `CMake3` is now supported by the `compile-cluster` script included in the `NDB` source distribution.

- 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`.

- 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.

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)

- **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.

- **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.

- **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.

- 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)

Bugs Fixed

- **Packaging:** Expected NDB header files were in the `devel` RPM package instead of `libndbclient-devel`. (Bug #84580, Bug #26448330)

- 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)

- **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`.

- **NDB Cluster APIs:** Added the `Table` methods `getExtraMetadata()` and `setExtraMetadata()`.

- `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).

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` 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)

- **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)

- 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 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

A

B

C

D

E

O

ODirect, 82, 156
ODirectSyncFlag, 82, 156
OM_WRITE_BUFFER, 82, 156
ON DELETE CASCADE, 69, 145
online DDL, 56, 133
online upgrades, 82, 156
OO_NOWAIT, 56, 133
optimizer, 64, 140
option handling, 82, 156
ORDER BY, 56, 82, 133, 156
OS X, 43, 122
O_DIRECT, 16, 29, 48, 97, 109, 125

P

Packaging, 10, 16, 29, 43, 64, 69, 79, 92, 97, 109, 122, 140, 145, 153
packaging, 38, 116
packed send, 23, 104
parallelism, 43, 69, 122, 145
partial LCP, 23, 48, 104, 125
partial restarts, 48, 125
PartionCount, 16, 97
Partitioning, 56, 133
PartitionsPerNode, 23, 104
PARTITION_BALANCE, 82, 156
Performance, 48, 79, 125, 153
performance, 64, 140
PGMAN, 23, 29, 48, 82, 104, 109, 125, 156
pgman_time_track_stats, 48, 125
PK reads, 29, 109
PortNumber, 16, 97
PreferIPVersion, 10, 92
processes table, 82, 156
pushdown, 64, 69, 140, 145
pushdown joins, 38, 116
pushed conditions, 69, 145
pushed joins, 3, 56, 69, 133, 145

Q

QEP_TAB, 82, 156
QMGR, 43, 122
query memory, 69, 145

R

race, 82, 156
race condition, 69, 145
race conditions, 64, 140
READ_BACKUP, 48, 125
RealtimeScheduler, 29, 109
receive thread, 23, 104
redo log, 10, 16, 69, 82, 92, 97, 145, 156
redo log part metadata, 82, 156
RedoOverCommitCounter, 48, 125

T

U

V

W