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 either the MySQL Forums or MySQL Mailing Lists, where you can discuss your issues with other MySQL users.

Document generated on: 2019-05-20 (revision: 17920)

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

Documentation Accessibility

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

Access to Oracle Support

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

Changes in MySQL NDB Cluster 8.0.17 (Not yet released)

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 (Not yet released, General Availability)).

Version 8.0.17-ndb-8.0.17 has no release notes, or they have not been published because the product version has not been released.

Changes in MySQL NDB Cluster 8.0.16 (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

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

- 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)).

  `ndb_restore` also now detects such a 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 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.
- 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 only the management server but no data nodes were started, RESTART ALL timed out and eventually failed. This was because, as part of a restart, ndb_mgmd starts a timer, sends a STOP_REQ signal to all the data nodes, and waits for all of them to reach node state SL_CMVMI. The issue arose becaue no STOP_REQ signals were ever sent, and thus no data nodes reached SL_CMVMI. This meant that the timer always expired, causing the restart to fail. (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)

- 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. (Bug #27484882)

- 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 acjknowledgements 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)

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:** MySQL NDB Cluster now implements schema synchronization of disk data objects including tablespaces and log file groups, 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.
- 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 NDB table extra metadata changes, and See also MySQL Data Dictionary.

Bugs Fixed

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

    This change 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`.

- `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 semi-join 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 semi-join, 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 semi-join 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 semi-join 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 8.0.16 (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

- 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)).

  `ndb_restore` also now detects such a 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 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.

- 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 only the management server but no data nodes were started, `RESTART ALL` timed out and eventually failed. This was because, as part of a restart, `ndb_mgmd` starts a timer, sends a `STOP_REQ` signal to all the data nodes, and waits for all of them to reach node state `SL_CMVMI`. The issue arose becaue no `STOP_REQ` signals were ever sent, and thus no data nodes reached `SL_CMVMI`. This meant that the timer always expired, causing the restart to fail. (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)

- 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. (Bug #27484882)

- 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 acjknowledgements 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)

Changes in MySQL NDB Cluster 8.0.14 (2019-01-21, Development Milestone)

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:** MySQL NDB Cluster now implements schema synchronization of disk data objects including tablespaces and log file groups, 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.
- 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 NDB table extra metadata changes, and See also MySQL Data Dictionary.

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

- **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

This change 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`.

- `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 semi-join 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 semi-join, 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 semi-join 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 semi-join 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

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W