MySQL NDB Cluster 7.4 Release Notes

Abstract

This document contains release notes for the changes in each release of MySQL NDB Cluster that uses version 7.4 of the `NDB` (`NDBCLUSTER`) storage engine.

Each NDB Cluster 7.4 release is based on a mainline MySQL Server release and a particular version of the `NDB` storage engine, as shown in the version string returned by executing `SELECT VERSION()` in the `mysql` client, or by executing the `ndb_mgm` client `SHOW` or `STATUS` command; for more information, see MySQL NDB Cluster 7.3 and NDB Cluster 7.4.

For general information about features added in NDB Cluster 7.4, see What is New in NDB Cluster. For a complete list of all bugfixes and feature changes in MySQL Cluster, please refer to the changelog section for each individual NDB Cluster release.

For additional MySQL 5.6 documentation, see the MySQL 5.6 Reference Manual, which includes an overview of features added in MySQL 5.6 that are not specific to NDB Cluster (What Is New in MySQL 5.6), and discussion of upgrade issues that you may encounter for upgrades from MySQL 5.5 to MySQL 5.6 (Changes Affecting Upgrades to MySQL 5.6). For a complete list of all bugfixes and feature changes made in MySQL 5.6 that are not specific to `NDB`, see MySQL 5.6 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 (http://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.

For additional documentation on MySQL products, including translations of the documentation into other languages, and downloadable versions in variety of formats, including HTML and PDF formats, see the MySQL Documentation Library.

Document generated on: 2017-01-04 (revision: 10541)

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 7.4 of the `NDB` storage engine.

Legal Notices

Documentation Accessibility

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

Access to Oracle Support

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 7.4.14 (5.6.34-ndb-7.4.14) (Not yet released, General Availability)

MySQL Cluster NDB 7.4.14 is a new release of MySQL Cluster NDB 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous MySQL Cluster releases.

Obtaining MySQL Cluster NDB 7.4.   MySQL Cluster NDB 7.4 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.4, see What is New in NDB Cluster 7.4.

This release also incorporates all bugfixes and changes made in previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.34 (see Changes in MySQL 5.6.34 (2016-10-12, General Availability)).

Version 5.6.34-ndb-7.4.14 has no changelog entries, or they have not been published because the product version has not been released.

Changes in MySQL NDB Cluster 7.4.13 (5.6.34-ndb-7.4.13) (2016-10-18, General Availability)

MySQL Cluster NDB 7.4.13 is a new release of MySQL Cluster NDB 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous MySQL Cluster releases.

Obtaining MySQL Cluster NDB 7.4.   MySQL Cluster NDB 7.4 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.4, see What is New in NDB Cluster 7.4.

This release also incorporates all bugfixes and changes made in previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.34 (see Changes in MySQL 5.6.34 (2016-10-12, General Availability)).

- Functionality Added or Changed

- Bugs Fixed

Functionality Added or Changed

- **MySQL NDB ClusterJ:** To help applications handle database errors better, a number of new features have been added to the `ClusterJDatastoreException` class:

  - A new method, `getCode()`, returns `code` from the `NdbError` object.

  - A new method, `getMysqlCode()`, returns `mysql_code` from the `NdbError` object.

  - A new subclass, `ClusterJDatastoreException.Classification`, gives users the ability to decode the result from `getClassification()`. The method `Classification.toString()` gives the name of the error classification as listed in NDB Error Classifications.

  (Bug #22353594)

Bugs Fixed

- Passing a nonexistent node ID to `CREATE NODEGROUP` led to random data node failures. (Bug #23748958)

- `DROP TABLE` followed by a node shutdown and subesequent master takeover—and with the containing local checkpoint not yet complete prior to the takeover—caused the LCP to be ignored, and in some cases, the data node to fail. (Bug #23735996)

  References: See also: Bug #23288252.

- Removed an invalid assertion to the effect that all cascading child scans are closed at the time API connection records are released following an abort of the main transaction. The assertion was invalid because closing of scans in such cases is by design asynchronous with respect to the main transaction, which means that subscans may well take some time to close after the main transaction is closed. (Bug #23709284)

- A number of potential buffer overflow issues were found and fixed in the `NDB` codebase. (Bug #23152979)

- A `SIGNAL_DROPPED_REP` handler invoked in response to long message buffer exhaustion was defined in the `SPJ` kernel block, but not actually used. This meant that the default handler from `SimulatedBlock` was used instead in such cases, which shut down the data node. (Bug #23048816)

  References: See also: Bug #23251145, Bug #23251423.

- When a data node has insufficient redo buffer during a system restart, it does not participate in the restart until after the other nodes have started. After this, it performs a takeover of its fragments from the nodes in its node group that have already started; during this time, the cluster is already running and user activity is possible, including DML and DDL operations.

  During a system restart, table creation is handled differently in the `DIH` kernel block than normally, as this creation actually consists of reloading table definition data from disk on the master node. Thus, `DIH` assumed that any table creation that occurred before all nodes had restarted must be related to the restart and thus always on the master node. However, during the takeover, table creation can occur on non-master nodes due to user activity; when this happened, the cluster underwent a forced shutdown.

Now an extra check is made during system restarts to detect in such cases whether the executing node is the master node, and use that information to determine whether the table creation is part of the restart proper, or is taking place during a subsequent takeover. (Bug #23028418)

- `ndb_restore` set the `MAX_ROWS` attribute for a table for which it had not been set prior to taking the backup. (Bug #22904640)

- Whenever data nodes are added to or dropped from the cluster, the `NDB` kernel's Event API is notified of this using a `SUB_GCP_COMPLETE_REP` signal with either the `ADD` (add) flag or `SUB` (drop) flag set, as well as the number of nodes to add or drop; this allows `NDB` to maintain a correct count of `SUB_GCP_COMPLETE_REP` signals pending for every incomplete bucket. In addition to handling the bucket for the epoch associated with the addition or removal, it must also compensate for any later incomplete buckets associated with later epochs. Although it was possible to complete such buckets out of order, there was no handling of these, leading a stall in to event reception.

  This fix adds detection and handling of such out of order bucket completion. (Bug #20402364)

  References: See also: Bug #82424, Bug #24399450.

- The count displayed by the `c_exec` column in the `ndbinfo.threadstat` table was incomplete. (Bug #82635, Bug #24482218)

- The internal function `ndbcluster_binlog_wait()`, which provides a way to make sure that all events originating from a given thread arrive in the binary log, is used by `SHOW BINLOG EVENTS` as well as when resetting the binary log. This function waits on an injector condition while the latest global epoch handled by `NDB` is more recent than the epoch last committed in this session, which implies that this condition must be signalled whenever the binary log thread completes and updates a new latest global epoch. Inspection of the code revealed that this condition signalling was missing, and that, instead of being awakened whenever a new latest global epoch completes (~100ms), client threads waited for the maximum timeout (1 second).

  This fix adds the missing injector condition signalling, while also changing it to a condition broadcast to make sure that all client threads are alerted. (Bug #82630, Bug #24481551)

- During a node restart, a fragment can be restored using information obtained from local checkpoints (LCPs); up to 2 restorable LCPs are retained at any given time. When an LCP is reported to the `DIH` kernel block as completed, but the node fails before the last global checkpoint index written into this LCP has actually completed, the latest LCP is not restorable. Although it should be possible to use the older LCP, it was instead assumed that no LCP existed for the fragment, which slowed the restart process. Now in such cases, the older, restorable LCP is used, which should help decrease long node restart times. (Bug #81894, Bug #23602217)

- While a `mysqld` was waiting to connect to the management server during initialization of the `NDB` handler, it was not possible to shut down the `mysqld`. If the `mysqld` was not able to make the connection, it could become stuck at this point. This was due to an internal wait condition in the utility and index statistics threads that could go unmet indefinitely. This condition has been augmented with a maximum timeout of 1 second, which makes it more likely that these threads terminate themselves properly in such cases.

  In addition, the connection thread waiting for the management server connection performed 2 sleeps in the case just described, instead of 1 sleep, as intended. (Bug #81585, Bug #23343673)

- The list of deferred tree node lookup requests created when preparing to abort a `DBSPJ` request were not cleared when this was complete, which could lead to deferred operations being started even after the `DBSPJ` request aborted. (Bug #81355, Bug #23251423)

  References: See also: Bug #23048816.

- Error and abort handling in `Dbspj::execTRANSID_AI()` was implemented such that its `abort()` method was called before processing of the incoming signal was complete. Since this method

sends signals to the LDM, this partly overwrote the contents of the signal which was later required by `execTRANSID_AI()`. This could result in aborted `DBSPJ` requests cleaning up their allocated resources too early, or not at all. (Bug #81353, Bug #23251145)

References: See also: Bug #23048816.

- Several object constructors and similar functions in the `NDB` codebase did not always perform sanity checks when creating new instances. These checks are now performed under such circumstances. (Bug #77408, Bug #21286722)

- An internal call to `malloc()` was not checked for `NULL`. The function call was replaced with a direct write. (Bug #77375, Bug #21271194)

- **NDB Cluster APIs:** Reuse of transaction IDs could occur when `Ndb` objects were created and deleted concurrently. As part of this fix, the NDB API methods `lock_ndb_objects()` and `unlock_ndb_objects` are now declared as `const`. (Bug #23709232)

- **NDB Cluster APIs:** When the management server was restarted while running an MGM API application that continuously monitored events, subsequent events were not reported to the application, with timeouts being returned indefinitely instead of an error.

  This occurred because sockets for event listeners were not closed when restarting `mgmd`. This is fixed by ensuring that event listener sockets are closed when the management server shuts down, causing applications using functions such as `ndb_logevent_get_next()` to receive a read error following the restart. (Bug #19474782)

Changes in MySQL NDB Cluster 7.4.12 (5.6.31-ndb-7.4.12) (2016-07-18, General Availability)

MySQL Cluster NDB 7.4.12 is a new release of MySQL Cluster NDB 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous MySQL Cluster releases.

Obtaining MySQL Cluster NDB 7.4.   MySQL Cluster NDB 7.4 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.4, see What is New in NDB Cluster 7.4.

This release also incorporates all bugfixes and changes made in previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.31 (see Changes in MySQL 5.6.31 (2016-06-02, General Availability)).

- Functionality Added or Changed

- Bugs Fixed

Functionality Added or Changed

- **MySQL NDB ClusterJ:** To make it easier for ClusterJ to handle fatal errors that require the `SessionFactory` to be closed, a new public method in the `SessionFactory` interface, getConnectionPoolSessionCounts(), has been created. When it returns zeros for all pooled connections, it means all sessions have been closed, at which point the `SessionFactory` can be closed and reopened. See Reconnecting to an NDB Cluster for more detail. (Bug #22353594)

Bugs Fixed

- **Incompatible Change:** When the data nodes are only partially connected to the API nodes, a node used for a pushdown join may get its request from a transaction coordinator on a different node, without (yet) being connected to the API node itself. In such cases, the `NodeInfo` object for the requesting API node contained no valid info about the software version of the API node, which caused the `DBSPJ` block to assume (incorrectly) when aborting to assume that the API node used

`NDB` version 7.2.4 or earlier, requiring the use of a backward compatability mode to be used during query abort which sent a node failure error instead of the real error causing the abort.

Now, whenever this situation occurs, it is assumed that, if the `NDB` software version is not yet available, the API node version is greater than 7.2.4. (Bug #23049170)

- Although arguments to the `DUMP` command are 32-bit integers, `ndb_mgmd` used a buffer of only 10 bytes when processing them. (Bug #23708039)

- During shutdown, the `mysqld` process could sometimes hang after logging `NDB Util: Stop` ... `NDB Util: Wakeup`. (Bug #23343739)

  References: See also: Bug #21098142.

- During an online upgrade from a MySQL Cluster NDB 7.3 release to an NDB 7.4 (or later) release, the failures of several data nodes running the lower version during local checkpoints (LCPs), and just prior to upgrading these nodes, led to additional node failures following the upgrade. This was due to lingering elements of the `EMPTY_LCP` protocol initiated by the older nodes as part of an LCP-plus-restart sequence, and which is no longer used in NDB 7.4 and later due to LCP optimizations implemented in those versions. (Bug #23129433)

- Reserved send buffer for the loopback transporter, introduced in MySQL Cluster NDB 7.4.8 and used by API and management nodes for administrative signals, was calculated incorrectly. (Bug #23093656, Bug #22016081)

  References: This issue is a regression of: Bug #21664515.

- During a node restart, re-creation of internal triggers used for verifying the referential integrity of foreign keys was not reliable, because it was possible that not all distributed TC and LDM instances agreed on all trigger identities. To fix this problem, an extra step is added to the node restart sequence, during which the trigger identities are determined by querying the current master node. (Bug #23068914)

  References: See also: Bug #23221573.

- Following the forced shutdown of one of the 2 data nodes in a cluster where `NoOfReplicas=2`, the other data node shut down as well, due to arbitration failure. (Bug #23006431)

- The `ndbinfo.tc_time_track_stats` table uses histogram buckets to give a sense of the distribution of latencies. The sizes of these buckets were also reported as `HISTOGRAM BOUNDARY INFO` messages during data node startup; this printout was redundant and so has been removed. (Bug #22819868)

- A failure occurred in `DBTUP` in debug builds when variable-sized pages for a fragment totalled more than 4 GB. (Bug #21313546)

- `mysqld` did not shut down cleanly when executing `ndb_index_stat`. (Bug #21098142)

  References: See also: Bug #23343739.

- `DBDICT` and `GETTABINFOREQ` queue debugging were enhanced as follows:

  - Monitoring by a data node of the progress of `GETTABINFOREQ` signals can be enabled by setting `DictTrace` >= 2.

  - Added the `ApiVerbose` configuration parameter, which enables NDB API debug logging for an API node where it is set greater than or equal to 2.

  - Added `DUMP` code 1229 which shows the current state of the `GETTABINFOREQ` queue. (See DUMP 1229.)

  See also The DBDICT Block. (Bug #20368450)

References: See also: Bug #20368354.

- **NDB Cluster APIs:** Deletion of Ndb objects used a dispoportionately high amount of CPU. (Bug #22986823)

Changes in MySQL NDB Cluster 7.4.11 (5.6.29-ndb-7.4.11) (2016-04-20, General Availability)

MySQL Cluster NDB 7.4.11 is a new release of MySQL Cluster NDB 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous MySQL Cluster releases.

Obtaining MySQL Cluster NDB 7.4.   MySQL Cluster NDB 7.4 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.4, see What is New in NDB Cluster 7.4.

This release also incorporates all bugfixes and changes made in previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.29 (see Changes in MySQL 5.6.29 (2016-02-05, General Availability)).

- Functionality Added or Changed

- Bugs Fixed

Functionality Added or Changed

- **NDB Cluster APIs:** Added the `Ndb::setEventBufferQueueEmptyEpoch()` method, which makes it possible to enable queuing of empty events (event type `TE_EMPTY`). (Bug #22157845)

Bugs Fixed

- **Important Change:** The minimum value for the `BackupDataBufferSize` data node configuration parameter has been lowered from 2 MB to 512 KB. The default and maximum values for this parameter remain unchanged. (Bug #22749509)

- **Microsoft Windows:** Performing `ANALYZE TABLE` on a table having one or more indexes caused `ndbmtd` to fail with an `InvalidAttrInfo` error due to signal corruption. This issue occurred consistently on Windows, but could also be encountered on other platforms. (Bug #77716, Bug #21441297)

- During node failure handling, the request structure used to drive the cleanup operation was not maintained correctly when the request was executed. This led to inconsistencies that were harmless during normal operation, but these could lead to assertion failures during node failure handling, with subsequent failure of additional nodes. (Bug #22643129)

- The previous fix for a lack of mutex protection for the internal `TransporterFacade::deliver_signal()` function was found to be incomplete in some cases. (Bug #22615274)

  References: This issue is a regression of: Bug #77225, Bug #21185585.

- Compilation of MySQL with Visual Studio 2015 failed in `ConfigInfo.cpp`, due to a change in Visual Studio's handling of spaces and concatenation. (Bug #22558836, Bug #80024)

- When setup of the binary log as an atomic operation on one SQL node failed, this could trigger a state in other SQL nodes in which they appeared to detect the SQL node participating in schema change distribution, whereas it had not yet completed binary log setup. This could in turn cause a deadlock on the global metadata lock when the SQL node still retrying binary log setup needed this

lock, while another mysqld had taken the lock for itself as part of a schema change operation. In such cases, the second SQL node waited for the first one to act on its schema distribution changes, which it was not yet able to do. (Bug #22494024)

- Duplicate key errors could occur when `ndb_restore` was run on a backup containing a unique index. This was due to the fact that, during restoration of data, the database can pass through one or more inconsistent states prior to completion, such an inconsistent state possibly having duplicate values for a column which has a unique index. (If the restoration of data is preceded by a run with `--disable-indexes` and followed by one with `--rebuild-indexes`, these errors are avoided.)

  Added a check for unique indexes in the backup which is performed only when restoring data, and which does not process tables that have explicitly been excluded. For each unique index found, a warning is now printed. (Bug #22329365)

- Restoration of metadata with `ndb_restore -m` occasionally failed with the error message `Failed to create index...` when creating a unique index. While disgnosing this problem, it was found that the internal error **PREPARE_SEIZE_ERROR** (a temporary error) was reported as an unknown error. Now in such cases, `ndb_restore` retries the creation of the unique index, and **PREPARE_SEIZE_ERROR** is reported as NDB Error 748 `Busy during read of event table`. (Bug #21178339)

  References: See also: Bug #22989944.

- When setting up event logging for `ndb_mgmd` on Windows, MySQL Cluster tries to add a registry key to `HKEY_LOCAL_MACHINE`, which fails if the user does not have access to the registry. In such cases `ndb_mgmd` logged the error `Could neither create or open key`, which is not accurate and which can cause confusion for users who may not realize that file logging is available and being used. Now in such cases, `ndb_mgmd` logs a warning `Could not create or access the registry key needed for the application to log to the Windows EventLog. Run the application with sufficient privileges once to create the key, or add the key manually, or turn off logging for that application`. An error (as opposed to a warning) is now reported in such cases only if there is no available output at all for `ndb_mgmd` event logging. (Bug #20960839)

- `NdbDictionary` metadata operations had a hard-coded 7-day timeout, which proved to be excessive for short-lived operations such as retrieval of table definitions. This could lead to unnecessary hangs in user applications which were difficult to detect and handle correctly. To help address this issue, timeout behaviour is modified so that read-only or short-duration dictionary interactions have a 2-minute timeout, while schema transactions of potentially long duration retain the existing 7-day timeout.

  Such timeouts are intended as a safety net: In the event of problems, these return control to users, who can then take corrective action. Any reproducible issue with `NdbDictionary` timeouts should be reported as a bug. (Bug #20368354)

- Optimization of signal sending by buffering and sending them periodically, or when the buffer became full, could cause `SUB_GCP_COMPLETE_ACK` signals to be excessively delayed. Such signals are sent for each node and epoch, with a minimum interval of `TimeBetweenEpochs`; if they are not received in time, the `SUMA` buffers can overflow as a result. The overflow caused API nodes to be disconnected, leading to current transactions being aborted due to node failure. This condition made it difficult for long transactions (such as altering a very large table), to be completed. Now in such cases, the `ACK` signal is sent without being delayed. (Bug #18753341)

- An internal function used to validate connections failed to update the connection count when creating a new `Ndb` object. This had the potential to create a new `Ndb` object for every operation validating the connection, which could have an impact on performance, particularly when performing schema operations. (Bug #80750, Bug #22932982)

- When an SQL node was started, and joined the schema distribution protocol, another SQL node, already waiting for a schema change to be distributed, timed out during that wait. This was because

the code incorrectly assumed that the new SQL node would also acknowledge the schema distribution even though the new node joined too late to be a participant in it.

As part of this fix, printouts of schema distribution progress now always print the more significant part of a bitmask before the less significant; formatting of bitmasks in such printouts has also been improved. (Bug #80554, Bug #22842538)

- Settings for the `SchedulerResponsiveness` data node configuration parameter (introduced in MySQL Cluster NDB 7.4.9) were ignored. (Bug #80341, Bug #22712481)

- MySQL Cluster did not compile correctly with Microsoft Visual Studio 2015, due to a change from previous versions in the VS implementation of the `_vsnprintf()` function. (Bug #80276, Bug #22670525)

- When setting CPU spin time, the value was needlessly cast to a boolean internally, so that setting it to any nonzero value yielded an effective value of 1. This issue, as well as the fix for it, apply both to setting the `SchedulerSpinTimer` parameter and to setting `spintime` as part of a `ThreadConfig` parameter value. (Bug #80237, Bug #22647476)

- Processing of local checkpoints was not handled correctly on Mac OS X, due to an uninitialized variable. (Bug #80236, Bug #22647462)

- A logic error in an `if` statement in `storage/ndb/src/kernel/blocks/dbacc/DbaccMain.cpp` rendered useless a check for determining whether `ZREAD_ERROR` should be returned when comparing operations. This was detected when compiling with `gcc` using `-Werror=logical-op`. (Bug #80155, Bug #22601798)

  References: This issue is a regression of: Bug #21285604.

- The `ndb_print_file` utility failed consistently on Solaris 9 for SPARC. (Bug #80096, Bug #22579581)

- Builds with the `-Werror` and `-Wextra` flags (as for release builds) failed on SLES 11. (Bug #79950, Bug #22539531)

- When using `CREATE INDEX` to add an index on either of two `NDB` tables sharing circular foreign keys, the query succeeded but a temporary table was left on disk, breaking the foreign key constraints. This issue was also observed when attempting to create an index on a table in the middle of a chain of foreign keys—that is, a table having both parent and child keys, but on different tables. The problem did not occur when using `ALTER TABLE` to perform the same index creation operation; and subsequent analysis revealed unintended differences in the way such operations were performed by `CREATE INDEX`.

  To fix this problem, we now make sure that operations performed by a `CREATE INDEX` statement are always handled internally in the same way and at the same time that the same operations are handled when performed by `ALTER TABLE` or `DROP INDEX`. (Bug #79156, Bug #22173891)

- `NDB` failed to ignore index prefixes on primary and unique keys, causing `CREATE TABLE` and `ALTER TABLE` statements using them to be rejected. (Bug #78441, Bug #21839248)

- **NDB Cluster APIs:** Executing a transaction with an `NdbIndexOperation` based on an obsolete unique index caused the data node process to fail. Now the index is checked in such cases, and if it cannot be used the transaction fails with an appropriate error. (Bug #79494, Bug #22299443)

- Integer overflow could occur during client handshake processing, leading to a server exit. (Bug #22722946)

- For busy servers, client connection or communication failure could occur if an I/O-related system call was interrupted. The `mysql_options()` C API function now has a `MYSQL_OPT_RETRY_COUNT` option to control the number of retries for interrupted system calls. (Bug #22336527)

  References: See also: Bug #22389653.

Changes in MySQL NDB Cluster 7.4.10 (5.6.28-ndb-7.4.10) (2016-01-29, General Availability)

MySQL Cluster NDB 7.4.10 is a new release of MySQL Cluster NDB 7.4 fixing a major regression in performance during restarts found in MySQL Cluster NDB 7.4.8 which also affected MySQL Cluster NDB 7.4.9. Users of previous releases of MySQL Cluster can and should bypass the 7.4.8 and 7.4.9 releases when performing an upgrade, and upgrade directly to MySQL Cluster NDB 7.4.10 or later.

Obtaining MySQL Cluster NDB 7.4. MySQL Cluster NDB 7.4 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.4, see What is New in NDB Cluster 7.4.

This release also incorporates all bugfixes and changes made in MySQL Cluster NDB 7.4.9 and previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.28 (see Changes in MySQL 5.6.28 (2015-12-07, General Availability)).

Bugs Fixed

- A serious regression was inadvertently introduced in MySQL Cluster NDB 7.4.8 whereby local checkpoints and thus restarts often took much longer than expected. This occurred due to the fact that the setting for `MaxDiskWriteSpeedOwnRestart` was ignored during restarts and the value of `MaxDiskWriteSpeedOtherNodeRestart`, which is much lower by default than the default for `MaxDiskWriteSpeedOwnRestart`, was used instead. This issue affected restart times and performance only and did not have any impact on normal operations. (Bug #22582233)

Changes in MySQL NDB Cluster 7.4.9 (5.6.28-ndb-7.4.9) (2016-01-18, General Availability)

Note

MySQL Cluster NDB 7.4.9 included a serious regression in performance during restarts, discovered shortly after release, and is replaced by MySQL Cluster NDB 7.4.10. Users of previous MySQL Cluster NDB 7.4 releases are advised to upgrade to MySQL Cluster NDB 7.4.10 or later, by passing NDB 7.4.9.

Obtaining MySQL Cluster NDB 7.4. MySQL Cluster NDB 7.4 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.4, see What is New in NDB Cluster 7.4.

This release also incorporates all bugfixes and changes made in previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.28 (see Changes in MySQL 5.6.28 (2015-12-07, General Availability)).

- Functionality Added or Changed

- Bugs Fixed

Functionality Added or Changed

- **Important Change:** Previously, the NDB scheduler always optimized for speed against throughput in a predetermined manner (this was hard coded); this balance can now be set using the `SchedulerResponsiveness` data node configuration parameter. This parameter accepts an integer in the range of 0-10 inclusive, with 5 as the default. Higher values provide better response times relative to throughput. Lower values provide increased throughput, but impose longer response times. (Bug #78531, Bug #21889312)

- Added the `tc_time_track_stats` table to the `ndbinfo` information database. This table provides time-tracking information relating to transactions, key operations, and scan operations performed by `NDB`. (Bug #78533, Bug #21889652)

- **NDB Replication:** Normally, `RESET SLAVE` causes all entries to be deleted from the `mysql.ndb_apply_status` table. This release adds the `ndb_clear_apply_status` system variable, which makes it possible to override this behavior. This variable is `ON` by default; setting it to `OFF` keeps `RESET SLAVE` from purging the `ndb_apply_status` table. (Bug #12630403)

Bugs Fixed

- **Important Change:** A fix made in MySQL Cluster NDB 7.3.11 and MySQL Cluster NDB 7.4.8 caused `ndb_restore` to perform unique key checks even when operating in modes which do not restore data, such as when using the program's `--restore_epoch` or `--print_data` option.

  That change in behavior caused existing valid backup routines to fail; to keep this issue from affecting this and future releases, the previous fix has been reverted. This means that the requirement added in those versions that ndb_restore be run `--disable-indexes` or `--rebuild-indexes` when used on tables containing unique indexes is also lifted. (Bug #22345748)

  References: See also: Bug #22329365. Reverted patches: Bug #57782, Bug #11764893.

- **Important Change:** Users can now set the number and length of connection timeouts allowed by most `NDB` programs with the `--connect-retries` and `--connect-retry-delay` command line options introduced for the programs in this release. For `ndb_mgm`, `--connect-retries` supersedes the existing `--try-reconnect` option. (Bug #57576, Bug #11764714)

- In debug builds, a `WAIT_EVENT` while polling caused excessive logging to stdout. (Bug #22203672)

- When executing a schema operation such as `CREATE TABLE` on a MySQL Cluster with multiple SQL nodes, it was possible for the SQL node on which the operation was performed to time out while waiting for an acknowledgement from the others. This could occur when different SQL nodes had different settings for `--ndb-log-updated-only`, `--ndb-log-update-as-write`, or other `mysqld` options effecting binary logging by `NDB`.

  This happened due to the fact that, in order to distribute schema changes between them, all SQL nodes subscribe to changes in the `ndb_schema` system table, and that all SQL nodes are made aware of each others subscriptions by subscribing to `TE_SUBSCRIBE` and `TE_UNSUBSCRIBE` events. The names of events to subscribe to are constructed from the table names, adding `REPL$` or `REPLF$` as a prefix. `REPLF$` is used when full binary logging is specified for the table. The issue described previously arose because different values for the options mentioned could lead to different events being subscribed to by different SQL nodes, meaning that all SQL nodes were not necessarily aware of each other, so that the code that handled waiting for schema distribution to complete did not work as designed.

  To fix this issue, MySQL Cluster now treats the `ndb_schema` table as a special case and enforces full binary logging at all times for this table, independent of any settings for `mysqld` binary logging options. (Bug #22174287, Bug #79188)

- Attempting to create an `NDB` table having greater than the maximum supported combined width for all `BIT` columns (4096) caused data node failure when these columns were defined with `COLUMN_FORMAT DYNAMIC`. (Bug #21889267)

- Creating a table with the maxmimum supported number of columns (512) all using `COLUMN_FORMAT DYNAMIC` led to data node failures. (Bug #21863798)

- In certain cases, a cluster failure (error 4009) was reported as `Unknown error code`. (Bug #21837074)

- For a timeout in `GET_TABINFOREQ` while executing a `CREATE INDEX` statement, mysqld returned Error 4243 (`Index not found`) instead of the expected Error 4008 (`Receive from NDB failed`).

The fix for this bug also fixes similar timeout issues for a number of other signals that are sent the `DBDICT` kernel block as part of DDL operations, including `ALTER_TAB_REQ`, `CREATE_INDX_REQ`, `DROP_FK_REQ`, `DROP_INDX_REQ`, `INDEX_STAT_REQ`, `DROP_FILE_REQ`, `CREATE_FILEGROUP_REQ`, `DROP_FILEGROUP_REQ`, `CREATE_EVENT`, `WAIT_GCP_REQ`, `DROP_TAB_REQ`, and `LIST_TABLES_REQ`, as well as several internal functions used in handling `NDB` schema operations. (Bug #21277472)

References: See also: Bug #20617891, Bug #20368354, Bug #19821115.

- Using `ndb_mgm STOP -f` to force a node shutdown even when it triggered a complete shutdown of the cluster, it was possible to lose data when a sufficient number of nodes were shut down, triggering a cluster shutodwn, and the timing was such that `SUMA` handovers had been made to nodes already in the process of shutting down. (Bug #17772138)

- The internal `NdbEventBuffer::set_total_buckets()` method calculated the number of remaining buckets incorrectly. This caused any incomplete epoch to be prematurely completed when the `SUB_START_CONF` signal arrived out of order. Any events belonging to this epoch arriving later were then ignored, and so effectively lost, which resulted in schema changes not being distributed correctly among SQL nodes. (Bug #79635, Bug #22363510)

- Compilation of MySQL Cluster failed on SUSE Linux Enterprise Server 12. (Bug #79429, Bug #22292329)

- Schema events were appended to the binary log out of order relative to non-schema events. This was caused by the fact that the binary log injector did not properly handle the case where schema events and non-schema events were from different epochs.

  This fix modifies the handling of events from the two schema and non-schema event streams such that events are now always handled one epoch at a time, starting with events from the oldest available epoch, without regard to the event stream in which they occur. (Bug #79077, Bug #22135584, Bug #20456664)

- When executed on an `NDB` table, `ALTER TABLE ... DROP INDEX` made changes to an internal array referencing the indexes before the index was actually dropped, and did not revert these changes in the event that the drop was not completed. One effect of this was that, after attempting to drop an index on which there was a foreign key dependency, the expected error referred to the wrong index, and subsequent attempts using SQL to modify indexes of this table failed. (Bug #78980, Bug #22104597)

- `NDB` failed during a node restart due to the status of the current local checkpoint being set but not as active, even though it could have other states under such conditions. (Bug #78780, Bug #21973758)

- `ndbmtd` checked for signals being sent only after a full cycle in `run_job_buffers`, which is performed for all job buffer inputs. Now this is done as part of `run_job_buffers` itself, which avoids executing for extended periods of time without sending to other nodes or flushing signals to other threads. (Bug #78530, Bug #21889088)

- The value set for `spintime` by the `ThreadConfig` parameter was not calculated correctly, causing the spin to continue for longer than actually specified. (Bug #78525, Bug #21886476)

- When `NDBFS` completed file operations, the method it employed for waking up the main thread worked effectively on Linux/x86 platforms, but not on some others, including OS X, which could lead to unnecessary slowdowns on those platforms. (Bug #78524, Bug #21886157)

- **NDB Disk Data:** A unique index on a column of an `NDB` table is implemented with an associated internal ordered index, used for scanning. While dropping an index, this ordered index was dropped first, followed by the drop of the unique index itself. This meant that, when the drop was rejected due to (for example) a constraint violation, the statement was rejected but the associated ordered index remained deleted, so that any subsequent operation using a scan on this table failed. We fix this problem by causing the unique index to be removed first, before removing the ordered index;

removal of the related ordered index is no longer performed when removal of a unique index fails. (Bug #78306, Bug #21777589)

- **NDB Replication:** While the binary log injector thread was handling failure events, it was possible for all `NDB` tables to be left indefinitely in read-only mode. This was due to a race condition between the binary log injector thread and the utility thread handling events on the ndb_schema table, and to the fact that, when handling failure events, the binary log injector thread places all `NDB` tables in read-only mode until all such events are handled and the thread restarts itself.

  When the binary log inject thread receives a group of one or more failure events, it drops all other existing event operations and expects no more events from the utility thread until it has handled all of the failure events and then restarted itself. However, it was possible for the utility thread to continue attempting binary log setup while the injector thread was handling failures and thus attempting to create the schema distribution tables as well as event subscriptions on these tables. If the creation of these tables and event subscriptions occurred during this time, the binary log injector thread's expectation that there were no further event operations was never met; thus, the injector thread never restarted, and `NDB` tables remained in read-only as described previously.

  To fix this problem, the `Ndb` object that handles schema events is now definitely dropped once the `ndb_schema` table drop event is handled, so that the utility thread cannot create any new events until after the injector thread has restarted, at which time, a new `Ndb` object for handling schema events is created. (Bug #17674771, Bug #19537961, Bug #22204186, Bug #22361695)

- **NDB Cluster APIs:** The binary log injector did not work correctly with `TE_INCONSISTENT` event type handling by `Ndb::nextEvent()`. (Bug #22135541)

  References: See also: Bug #20646496.

- **NDB Cluster APIs:** `Ndb::pollEvents()` and `pollEvents2()` were slow to receive events, being dependent on other client threads or blocks to perform polling of transporters on their behalf. This fix allows a client thread to perform its own transporter polling when it has to wait in either of these methods.

  Introduction of transporter polling also revealed a problem with missing mutex protection in the `ndbcluster_binlog` handler, which has been added as part of this fix. (Bug #79311, Bug #20957068, Bug #22224571)

- **NDB Cluster APIs:** Garbage collection is performed on several objects in the implementation of `NdbEventOperation`, based on which GCIs have been consumed by clients, including those that have been dropped by `Ndb::dropEventOperation()`. In this implementation, the assumption was made that the global checkpoint index (GCI) is always monotonically increasing, although this is not the case during an initial restart, when the GCI is reset. This could lead to event objects in the NDB API being released prematurely or not at all, in the latter case causing a resource leak.

  To prevent this from happening, the NDB event object's implementation now tracks, internally, both the GCI and the generation of the GCI; the generation is incremented whenever the node process is restarted, and this value is now used to provide a monotonically increasing sequence. (Bug #73781, Bug #21809959)

Changes in MySQL NDB Cluster 7.4.8 (5.6.27-ndb-7.4.8) (2015-10-16, General Availability)

MySQL Cluster NDB 7.4.8 is a new release of MySQL Cluster NDB 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous MySQL Cluster releases.

Obtaining MySQL Cluster NDB 7.4.   MySQL Cluster NDB 7.4 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.4, see What is New in NDB Cluster 7.4.

This release also incorporates all bugfixes and changes made in previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.27 (see Changes in MySQL 5.6.27 (2015-09-30, General Availability)).

- Functionality Added or Changed

- Bugs Fixed

Functionality Added or Changed

- **Incompatible Change:** The changes listed here follow up and build further on work done in MySQL Cluster NDB 7.4.7 to improve handling of local checkpoints (LCPs) under conditions of insert overload:

  - Changes have been made in the minimum values for a number of parameters applying to data buffers for backups and LCPs. These parameters, listed here, can no longer be set so as to make the system impossible to run:

    - `BackupDataBufferSize`: minimum increased from 0 to 2M.

    - `BackupLogBufferSize`: minimum increased from 0 to 2M.

    - `BackupWriteSize`: minimum increased from 2K to 32K.

    - `BackupMaxWriteSize`: minimum increased from 2K to 256K.

    In addition, the `BackupMemory` data node parameter is now deprecated and subject to removal in a future version of MySQL Cluster. Use `BackupDataBufferSize` and `BackupLogBufferSize` instead.

  - When a backup was unsuccessful due to insufficient resources, a subsequent retry worked only for those parts of the backup that worked in the same thread, since delayed signals are only supported in the same thread. Delayed signals are no longer sent to other threads in such cases.

  - An instance of an internal list object used in searching for queued scans was not actually destroyed before calls to functions that could manipulate the base object used to create it.

  - ACC scans were queued in the category of range scans, which could lead to starting an ACC scan when `DBACC` had no free slots for scans. We fix this by implementing a separate queue for ACC scans.

  (Bug #76890, Bug #20981491, Bug #77597, Bug #21362758, Bug #77612, Bug #21370839)

  References: See also: Bug #76742, Bug #20904721.

- **Important Change; NDB Replication:** Added the `create_old_temporals` server system variable to compliment the system variables `avoid_temporal_upgrade` and `show_old_temporals` introduced in MySQL 5.6.24 and available in MySQL Cluster beginning with NDB 7.3.9 and NDB 7.4.6. Enabling `create_old_temporals` causes `mysqld` to use the storage format employed prior to MySQL 5.6.4 when creating any `DATE`, `DATETIME`, or `TIMESTAMP` column—that is, the column is created without any support for fractional seconds. `create_old_temporals` is disabled by default. The system variable is read-only; to enable the use of pre-5.6.4 temporal types, set the equivalent option (`--create-old-temporals`) on the command line, or in an option file read by the MySQL server.

  `create_old_temporals` is available only in MySQL Cluster; it is not supported in the standard MySQL 5.6 server. It is intended to facilitate upgrades from MySQL Cluster NDB 7.2 to MySQL Cluster NDB 7.3 and 7.4, after which table columns of the affected types can be upgraded to the new storage format. `create_old_temporals` is deprecated and scheduled for removal in a future version of MySQL Cluster.

`avoid_temporal_upgrade` must also be enabled for this feature to work properly. You should also enable `show_old_temporals` as well. For more information, see the descriptions of these variables. For more about the changes in MySQL's temporal types, see Storage Requirements for Date and Time Types. (Bug #20701918)

References: See also: Bug #21492598, Bug #72997, Bug #18985760.

- When the `--database` option has not been specified for `ndb_show_tables`, and no tables are found in the `TEST_DB` database, an appropriate warning message is now issued. (Bug #50633, Bug #11758430)

Bugs Fixed

- **Important Change:** When `ndb_restore` was run without `--disable-indexes` or `--rebuild-indexes` on a table having a unique index, it was possible for rows to be restored in an order that resulted in duplicate values, causing it to fail with duplicate key errors. Running `ndb_restore` on such a table now requires using at least one of these options; failing to do so now results in an error. (Bug #57782, Bug #11764893)

  References: See also: Bug #22329365, Bug #22345748.

- **Important Change; NDB Cluster APIs:** The MGM API error-handling functions `ndb_mgm_get_latest_error()`, `ndb_mgm_get_latest_error_msg()`, and `ndb_mgm_get_latest_error_desc()` each failed when used with a `NULL` handle. You should note that, although these functions are now null-safe, values returned in this case are arbitrary and not meaningful. (Bug #78130, Bug #21651706)

- `mysql_upgrade` failed when performing an upgrade from MySQL Cluster NDB 7.2 to MySQL Cluster NDB 7.4. The root cause of this issue was an accidental duplication of code in `mysql_fix_privilege_tables.sql` that caused `ndbinfo_offline` mode to be turned off too early, which in turn led a subsequent `CREATE VIEW` statement to fail. (Bug #21841821)

- `ClusterMgr` is a internal component of NDB API and `ndb_mgmd` processes, part of `TransporterFacade`—which in turn is a wrapper around the transporter registry—and shared with data nodes. This component is responsible for a number of tasks including connection setup requests; sending and monitoring of heartbeats; provision of node state information; handling of cluster disconnects and reconnects; and forwarding of cluster state indicators. `ClusterMgr` maintains a count of live nodes which is incremented on receiving a report of a node having connected (`reportConnected()` method call), and decremented on receiving a report that a node has disconnected (`reportDisconnected()`) from `TransporterRegistry`. This count is checked within `reportDisconnected()` to verify that is it greater than zero.

  The issue addressed here arose when node connections were very brief due to send buffer exhaustion (among other potential causes) and the check just described failed. This occurred because, when a node did not fully connect, it was still possible for the connection attempt to trigger a `reportDisconnected()` call in spite of the fact that the connection had not yet been reported to `ClusterMgr`; thus, the pairing of `reportConnected()` and `reportDisconnected()` calls was not guaranteed, which could cause the count of connected nodes to be set to zero even though there remained nodes that were still in fact connected, causing node crashes with debug builds of MySQL Cluster, and potential errors or other adverse effects with release builds.

  To fix this issue, `ClusterMgr::reportDisconnected()` now verifies that a disconnected node had actually finished connecting completely before checking and decrementing the number of connected nodes. (Bug #21683144, Bug #22016081)

  References: See also: Bug #21664515, Bug #21651400.

- To reduce the possibility that a node's loopback transporter becomes disconnected from the transporter registry by `reportError()` due to send buffer exhaustion (implemented by the fix for

Bug #21651400), a portion of the send buffer is now reserved for the use of this transporter. (Bug #21664515, Bug #22016081)

References: See also: Bug #21651400, Bug #21683144.

- The loopback transporter is similar to the TCP transporter, but is used by a node to send signals to itself as part of many internal operations. Like the TCP transporter, it could be disconnected due to certain conditions including send buffer exhaustion, but this could result in blocking of `TransporterFacade` and so cause multiple issues within an `ndb_mgmd` or API node process. To prevent this, a node whose loopback transporter becomes disconnected is now simply shut down, rather than allowing the node process to hang. (Bug #21651400, Bug #22016081)

  References: See also: Bug #21683144, Bug #21664515.

- The internal `NdbEventBuffer` object's active subscriptions count (`m_active_op_count`) could be decremented more than once when stopping a subscription when this action failed, for example, due to a busy server and was retried. Decrementing of this count could also fail when communication with the data node failed, such as when a timeout occurred. (Bug #21616263)

  References: This issue is a regression of: Bug #20575424, Bug #20561446.

- In some cases, the management server daemon failed on startup without reporting the reason. Now when `ndb_mgmd` fails to start due to an error, the error message is printed to `stderr`. (Bug #21571055)

- In a MySQL Cluster with multiple LDM instances, all instances wrote to the node log, even inactive instances on other nodes. During restarts, this caused the log to be filled with messages from other nodes, such as the messages shown here:

```
2015-06-24 00:20:16 [ndbd] INFO     -- We are adjusting Max Disk Write Speed,
a restart is ongoing now
...
2015-06-24 01:08:02 [ndbd] INFO     -- We are adjusting Max Disk Write Speed,
no restarts ongoing anymore
```

  Now this logging is performed only by the active LDM instance. (Bug #21362380)

- Backup block states were reported incorrectly during backups. (Bug #21360188)

  References: See also: Bug #20204854, Bug #21372136.

- Added the `BackupDiskWriteSpeedPct` data node parameter. Setting this parameter causes the data node to reserve a percentage of its maximum write speed (as determined by the value of `MaxDiskWriteSpeed`) for use in local checkpoints while performing a backup. `BackupDiskWriteSpeedPct` is interpreted as a percentage which can be set between 0 and 90 inclusive, with a default value of 50. (Bug #20204854)

  References: See also: Bug #21372136.

- When a data node is known to have been alive by other nodes in the cluster at a given global checkpoint, but its `sysfile` reports a lower GCI, the higher GCI is used to determine which global checkpoint the data node can recreate. This caused problems when the data node being started had a clean file system (GCI = 0), or when it was more than more global checkpoint behind the other nodes.

  Now in such cases a higher GCI known by other nodes is used only when it is at most one GCI ahead. (Bug #19633824)

  References: See also: Bug #20334650, Bug #21899993. This issue is a regression of: Bug #29167.

- When restoring a specific database or databases with the `--include-databases` or `--exclude-databases` option, `ndb_restore` attempted to apply foreign keys on tables in databases which were not among those being restored. (Bug #18560951)

- After restoring the database schema from backup using `ndb_restore`, auto-discovery of restored tables in transactions having multiple statements did not work correctly, resulting in `Deadlock found when trying to get lock; try restarting transaction` errors.

  This issue was encountered both in the `mysql` client, as well as when such transactions were executed by application programs using Connector/J and possibly other MySQL APIs.

  Prior to upgrading, this issue can be worked around by executing `SELECT TABLE_NAME, TABLE_SCHEMA FROM INFORMATION_SCHEMA.TABLES WHERE ENGINE = 'NDBCLUSTER'` on all SQL nodes following the restore operation, before executing any other statements. (Bug #18075170)

- `ndb_desc` used with the `--extra-partition-info` and `--blob-info` options failed when run against a table containing one or more `TINYBLOB`. columns. (Bug #14695968)

- Operations relating to global checkpoints in the internal event data buffer could sometimes leak memory. (Bug #78205, Bug #21689380)

  References: See also: Bug #76165, Bug #20651661.

- Trying to create an `NDB` table with a composite foreign key referencing a composite primary key of the parent table failed when one of the columns in the composite foreign key was the table's primary key and in addition this column also had a unique key. (Bug #78150, Bug #21664899)

- When attempting to enable index statistics, creation of the required system tables, events and event subscriptions often fails when multiple `mysqld` processes using index statistics are started concurrently in conjunction with starting, restarting, or stopping the cluster, or with node failure handling. This is normally recoverable, since the affected `mysqld` process or processes can (and do) retry these operations shortly thereafter. For this reason, such failures are no longer logged as warnings, but merely as informational events. (Bug #77760, Bug #21462846)

- Adding a unique key to an `NDB` table failed when the table already had a foreign key. Prior to upgrading, you can work around this issue by creating the unique key first, then adding the foreign key afterwards, using a separate `ALTER TABLE` statement. (Bug #77457, Bug #20309828)

- **NDB Replication:** When using conflict detection and resolution with `NDB$EPOCH2_TRANS()`, delete-delete conflicts were not handled in a transactional manner. (Bug #20713499)

- **NDB Cluster APIs:** While executing `dropEvent()`, if the coordinator `DBDICT` failed after the subscription manager (`SUMA` block) had removed all subscriptions but before the coordinator had deleted the event from the system table, the dropped event remained in the table, causing any subsequent drop or create event with the same name to fail with `NDB` error 1419 `Subscription already dropped` or error 746 `Event name already exists`. This occurred even when calling `dropEvent()` with a nonzero force argument.

  Now in such cases, error 1419 is ignored, and `DBDICT` deletes the event from the table. (Bug #21554676)

- **NDB Cluster APIs:** If the total amount of memory allocated for the event buffer exceeded approximately 40 MB, the calculation of memory usage percentages could overflow during computation. This was due to the fact that the associated routine used 32-bit arithmetic; this has now been changed to use `Uint64` values instead. (Bug #78454, Bug #21847552)

- **NDB Cluster APIs:** The `nextEvent2()` method continued to return exceptional events such as `TE_EMPTY`, `TE_INCONSISTENT`, and `TE_OUT_OF_MEMORY` for event operations which already had been dropped. (Bug #78167, Bug #21673318)

- **NDB Cluster APIs:** After the initial restart of a node following a cluster failure, the cluster failure event added as part of the restart process was deleted when an event that existed prior to the restart was later deleted. This meant that, in such cases, an Event API client had no way of knowing that failure handling was needed. In addition, the GCI used for the final cleanup of deleted event operations, performed by `pollEvents()` and `nextEvent()` when these methods have consumed all available events, was lost. (Bug #78143, Bug #21660947)

- **NDB Cluster APIs:** The internal value representing the latest global checkpoint was not always updated when a completed epoch of event buffers was inserted into the event queue. This caused subsequent calls to `Ndb::pollEvents()` and `pollEvents2()` to fail when trying to obtain the correct GCI for the events available in the event buffers. This could also result in later calls to `nextEvent()` or `nextEvent2()` seeing events that had not yet been discovered. (Bug #78129, Bug #21651536)

Changes in MySQL NDB Cluster 7.4.7 (5.6.25-ndb-7.4.7) (2015-07-13, General Availability)

MySQL Cluster NDB 7.4.7 is a new release of MySQL Cluster NDB 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous MySQL Cluster releases.

Obtaining MySQL Cluster NDB 7.4.   MySQL Cluster NDB 7.4 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.4, see What is New in NDB Cluster 7.4.

This release also incorporates all bugfixes and changes made in previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.25 (see Changes in MySQL 5.6.25 (2015-05-29, General Availability)).

- Functionality Added or Changed

- Bugs Fixed

Functionality Added or Changed

- Deprecated MySQL Cluster node configuration parameters are now indicated as such by `ndb_config --configinfo --xml`. For each parameter currently deprecated, the corresponding `<param/>` tag in the XML output now includes the attribute `deprecated="true"`. (Bug #21127135)

- A number of improvements, listed here, have been made with regard to handling issues that could arise when an overload arose due to a great number of inserts being performed during a local checkpoint (LCP):

  - Failures sometimes occurred during restart processing when trying to execute the undo log, due to a problem with finding the end of the log. This happened when there remained unwritten pages at the end of the first undo file when writing to the second undo file, which caused the execution of undo logs in reverse order and so execute old or even nonexistent log records.

    This is fixed by ensuring that execution of the undo log begins with the proper end of the log, and, if started earlier, that any unwritten or faulty pages are ignored.

  - It was possible to fail during an LCP, or when performing a `COPY_FRAGREQ`, due to running out of operation records. We fix this by making sure that LCPs and `COPY_FRAG` use resources reserved for operation records, as was already the case with scan records. In addition, old code for ACC operations that was no longer required but that could lead to failures was removed.

  - When an LCP was performed while loading a table, it was possible to hit a livelock during LCP scans, due to the fact that that each record that was inserted into new pages after the LCP had started had its `LCP_SKIP` flag set. Such records were discarded as intended by the LCP scan, but

when inserts occurred faster than the LCP scan could discard records, the scan appeared to hang. As part of this issue, the scan failed to report any progress to the LCP watchdog, which after 70 seconds of livelock killed the process. This issue was observed when performing on the order of 250000 inserts per second over an extended period of time (120 seconds or more), using a single LDM.

This part of the fix makes a number of changes, listed here:

- We now ensure that pages created after the LCP has started are not included in LCP scans; we also ensure that no records inserted into those pages have their `LCP_SKIP` flag set.

- Handling of the scan protocol is changed such that a certain amount of progress is made by the LCP regardless of load; we now report progress to the LCP watchdog so that we avoid failure in in the event that an LCP is making progress but not writing any records.

- We now take steps to guarantee that LCP scans proceed more quickly than inserts can occur, by ensuring that scans are prioritized this scanning activity, and thus, that the LCP is in fact (eventually) completed.

- In addition, scanning is made more efficient, by prefetching tuples; this helps avoid stalls while fetching memory in the CPU.

- Row checksums for preventing data corruption now include the tuple header bits.

(Bug #76373, Bug #20727343, Bug #76741, Bug #69994, Bug #20903880, Bug #76742, Bug #20904721, Bug #76883, Bug #20980229)

- **MySQL NDB ClusterJ:** Under high workload, it was possible to overload the direct memory used to back domain objects, because direct memory is not garbage collected in the same manner as objects allocated on the heap. Two strategies have been added to the ClusterJ implementation: first, direct memory is now pooled, so that when the domain object is garbage collected, the direct memory can be reused by another domain object. Additionally, a new user-level method, `release(instance)`, has been added to the `Session` interface, which allows users to release the direct memory before the corresponding domain object is garbage collected. See the description for release(T) for more information. (Bug #20504741)

Bugs Fixed

- **Incompatible Change; NDB Cluster APIs:** The `pollEvents2()` method now returns -1, indicating an error, whenever a negative value is used for the time argument. (Bug #20762291)

- **Important Change; NDB Cluster APIs:** The `Ndb::getHighestQueuedEpoch()` method returned the greatest epoch in the event queue instead of the greatest epoch found after calling `pollEvents2()`. (Bug #20700220)

- **Important Change; NDB Cluster APIs:** `Ndb::pollEvents()` is now compatible with the `TE_EMPTY`, `TE_INCONSISTENT`, and `TE_OUT_OF_MEMORY` event types introduced in MySQL Cluster NDB 7.4.3. For detailed information about this change, see the description of this method in the *MySQL Cluster API Developer Guide*. (Bug #20646496)

- **Important Change; NDB Cluster APIs:** Added the method `Ndb::isExpectingHigherQueuedEpochs()` to the NDB API to detect when additional, newer event epochs were detected by `pollEvents2()`.

  The behavior of `Ndb::pollEvents()` has also been modified such that it now returns `NDB_FAILURE_GCI` (equal to `~(Uint64) 0`) when a cluster failure has been detected. (Bug #18753887)

- After restoring the database metadata (but not any data) by running `ndb_restore --restore_meta` (or `-m`), SQL nodes would hang while trying to `SELECT` from a table in the database to which the metadata was restored. In such cases the attempt to query the table now

fails as expected, since the table does not actually exist until `ndb_restore` is executed with `--restore_data` (`-r`). (Bug #21184102)

References: See also: Bug #16890703.

- When a great many threads opened and closed blocks in the NDB API in rapid succession, the internal `close_clnt()` function synchronizing the closing of the blocks waited an insufficiently long time for a self-signal indicating potential additional signals needing to be processed. This led to excessive CPU usage by `ndb_mgmd`, and prevented other threads from opening or closing other blocks. This issue is fixed by changing the function polling call to wait on a specific condition to be woken up (that is, when a signal has in fact been executed). (Bug #21141495)

- Previously, multiple send threads could be invoked for handling sends to the same node; these threads then competed for the same send lock. While the send lock blocked the additional send threads, work threads could be passed to other nodes.

  This issue is fixed by ensuring that new send threads are not activated while there is already an active send thread assigned to the same node. In addition, a node already having an active send thread assigned to it is no longer visible to other, already active, send threads; that is, such a node is longer added to the node list when a send thread is currently assigned to it. (Bug #20954804, Bug #76821)

- Queueing of pending operations when the redo log was overloaded (`DefaultOperationRedoProblemAction` API node configuration parameter) could lead to timeouts when data nodes ran out of redo log space (`P_TAIL_PROBLEM` errors). Now when the redo log is full, the node aborts requests instead of queuing them. (Bug #20782580)

  References: See also: Bug #20481140.

- An `NDB` event buffer can be used with an `Ndb` object to subscribe to table-level row change event streams. Users subscribe to an existing event; this causes the data nodes to start sending event data signals (`SUB_TABLE_DATA`) and epoch completion signals (`SUB_GCP_COMPLETE`) to the `Ndb` object. `SUB_GCP_COMPLETE_REP` signals can arrive for execution in concurrent receiver thread before completion of the internal method call used to start a subscription.

  Execution of `SUB_GCP_COMPLETE_REP` signals depends on the total number of `SUMA` buckets (sub data streams), but this may not yet have been set, leading to the present issue, when the counter used for tracking the `SUB_GCP_COMPLETE_REP` signals (`TOTAL_BUCKETS_INIT`) was found to be set to erroneous values. Now `TOTAL_BUCKETS_INIT` is tested to be sure it has been set correctly before it is used. (Bug #20575424, Bug #76255)

  References: See also: Bug #20561446, Bug #21616263.

- `NDB` statistics queries could be delayed by the error delay set for `ndb_index_stat_option` (default 60 seconds) when the index that was queried had been marked with internal error. The same underlying issue could also cause `ANALYZE TABLE` to hang when executed against an `NDB` table having multiple indexes where an internal error occured on one or more but not all indexes.

  Now in such cases, any existing statistics are returned immediately, without waiting for any additonal statistics to be discovered. (Bug #20553313, Bug #20707694, Bug #76325)

- The multi-threaded scheduler sends to remote nodes either directly from each worker thread or from dedicated send threadsL, depending on the cluster's configuration. This send might transmit all, part, or none of the available data from the send buffers. While there remained pending send data, the worker or send threads continued trying to send in a loop. The actual size of the data sent in the most recent attempt to perform a send is now tracked, and used to detect lack of send progress by the send or worker threads. When no progress has been made, and there is no other work outstanding, the scheduler takes a 1 millisecond pause to free up the CPU for use by other threads. (Bug #18390321)

  References: See also: Bug #20929176, Bug #20954804.

- In some cases, attempting to restore a table that was previously backed up failed with a `File Not Found` error due to a missing table fragment file. This occurred as a result of the NDB kernel `BACKUP` block receiving a `Busy` error while trying to obtain the table description, due to other traffic from external clients, and not retrying the operation.

  The fix for this issue creates two separate queues for such requests—one for internal clients such as the `BACKUP` block or `ndb_restore`, and one for external clients such as API nodes—and prioritizing the internal queue.

  Note that it has always been the case that external client applications using the NDB API (including MySQL applications running against an SQL node) are expected to handle `Busy` errors by retrying transactions at a later time; this expectation is *not* changed by the fix for this issue. (Bug #17878183)

  References: See also: Bug #17916243.

- On startup, API nodes (including `mysqld` processes running as SQL nodes) waited to connect with data nodes that had not yet joined the cluster. Now they wait only for data nodes that have actually already joined the cluster.

  In the case of a new data node joining an existing cluster, API nodes still try to connect with the new data node within `HeartbeatIntervalDbApi` milliseconds. (Bug #17312761)

- In some cases, the `DBDICT` block failed to handle repeated `GET_TABINFOREQ` signals after the first one, leading to possible node failures and restarts. This could be observed after setting a sufficiently high value for `MaxNoOfExecutionThreads` and low value for `LcpScanProgressTimeout`. (Bug #77433, Bug #21297221)

- Client lookup for delivery of API signals to the correct client by the internal `TransporterFacade::deliver_signal()` function had no mutex protection, which could cause issues such as timeouts encountered during testing, when other clients connected to the same `TransporterFacade`. (Bug #77225, Bug #21185585)

- It was possible to end up with a lock on the send buffer mutex when send buffers became a limiting resource, due either to insufficient send buffer resource configuration, problems with slow or failing communications such that all send buffers became exhausted, or slow receivers failing to consume what was sent. In this situation worker threads failed to allocate send buffer memory for signals, and attempted to force a send in order to free up space, while at the same time the send thread was busy trying to send to the same node or nodes. All of these threads competed for taking the send buffer mutex, which resulted in the lock already described, reported by the watchdog as `Stuck in Send`. This fix is made in two parts, listed here:

  1. The send thread no longer holds the global send thread mutex while getting the send buffer mutex; it now releases the global mutex prior to locking the send buffer mutex. This keeps worker threads from getting stuck in send in such cases.

  2. Locking of the send buffer mutex done by the send threads now uses a try-lock. If the try-lock fails, the node to make the send to is reinserted at the end of the list of send nodes in order to be retried later. This removes the `Stuck in Send` condition for the send threads.

  (Bug #77081, Bug #21109605)

- **NDB Cluster APIs:** Added the `Column::getSizeInBytesForRecord()` method, which returns the size required for a column by an `NdbRecord`, depending on the column's type (text/blob, or other). (Bug #21067283)

- **NDB Cluster APIs:** `NdbEventOperation::isErrorEpoch()` incorrectly returned `false` for the `TE_INCONSISTENT` table event type (see Event::TableEvent). This caused a subsequent call to `getEventType()` to fail. (Bug #20729091)

- **NDB Cluster APIs:** Creation and destruction of `Ndb_cluster_connection` objects by multiple threads could make use of the same application lock, which in some cases led to failures in the

global dictionary cache. To alleviate this problem, the creation and destruction of several internal NDB API objects have been serialized. (Bug #20636124)

- **NDB Cluster APIs:** A number of timeouts were not handled correctly in the NDB API.

  (Bug #20617891)

- **NDB Cluster APIs:** When an `Ndb` object created prior to a failure of the cluster was reused, the event queue of this object could still contain data node events originating from before the failure. These events could reference "old" epochs (from before the failure occurred), which in turn could violate the assumption made by the `nextEvent()` method that epoch numbers always increase. This issue is addressed by explicitly clearing the event queue in such cases. (Bug #18411034)

  References: See also: Bug #20888668.

- **MySQL NDB ClusterJ:** When used with Java 1.7 or higher, ClusterJ might cause the Java VM to crash when querying tables with BLOB columns, because `NdbDictionary::createRecord` calculates the wrong size needed for the record. Subsequently, when ClusterJ called `NdbScanOperation::nextRecordCopyOut`, the data overran the allocated buffer space. With this fix, ClusterJ checks the size calculated by `NdbDictionary::createRecord` and uses the value for the buffer size, if it is larger than the value ClusterJ itself calculates. (Bug #20695155)

Changes in MySQL NDB Cluster 7.4.6 (5.6.24-ndb-7.4.6) (2015-04-14, General Availability)

MySQL Cluster NDB 7.4.6 is a new release of MySQL Cluster NDB 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous MySQL Cluster releases.

Obtaining MySQL Cluster NDB 7.4. MySQL Cluster NDB 7.4 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.4, see What is New in NDB Cluster 7.4.

This release also incorporates all bugfixes and changes made in previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.24 (see Changes in MySQL 5.6.24 (2015-04-06, General Availability)).

Bugs Fixed

- During backup, loading data from one SQL node followed by repeated `DELETE` statements on the tables just loaded from a different SQL node could lead to data node failures. (Bug #18949230)

- When an instance of `NdbEventBuffer` was destroyed, any references to GCI operations that remained in the event buffer data list were not freed. Now these are freed, and items from the event bufer data list are returned to the free list when purging GCI containers. (Bug #76165, Bug #20651661)

- When a bulk delete operation was committed early to avoid an additional round trip, while also returning the number of affected rows, but failed with a timeout error, an SQL node performed no verification that the transaction was in the Committed state. (Bug #74494, Bug #20092754)

  References: See also: Bug #19873609.

Changes in MySQL NDB Cluster 7.4.5 (5.6.23-ndb-7.4.5) (2015-03-20, General Availability)

MySQL Cluster NDB 7.4.5 is a new release of MySQL Cluster NDB 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous MySQL Cluster releases.

Obtaining MySQL Cluster NDB 7.4.   MySQL Cluster NDB 7.4 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.4, see What is New in NDB Cluster 7.4.

This release also incorporates all bugfixes and changes made in previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.23 (see Changes in MySQL 5.6.23 (2015-02-02, General Availability)).

Bugs Fixed

- **Important Change:** The maximum failure time calculation used to ensure that normal node failure handling mechanisms are given time to handle survivable cluster failures (before global checkpoint watchdog mechanisms start to kill nodes due to GCP delays) was excessively conservative, and neglected to consider that there can be at most *number_of_data_nodes* / `NoOfReplicas` node failures before the cluster can no longer survive. Now the value of `NoOfReplicas` is properly taken into account when performing this calculation.

  This fix adds the `TimeBetweenGlobalCheckpointsTimeout` data node configuration parameter, which makes the minimum timeout between global checkpoints settable by the user. This timeout was previously fixed internally at 120000 milliseconds, which is now the default value for this parameter. (Bug #20069617, Bug #20069624)

  References: See also: Bug #19858151, Bug #20128256, Bug #20135976.

- In the event of a node failure during an initial node restart followed by another node start, the restart of the affected node could hang with a `START_INFOREQ` that occurred while invalidation of local checkpoints was still ongoing. (Bug #20546157, Bug #75916)

  References: See also: Bug #34702.

- It was found during testing that problems could arise when the node registered as the arbitrator disconnected or failed during the arbitration process.

  In this situation, the node requesting arbitration could never receive a positive acknowledgement from the registered arbitrator; this node also lacked a stable set of members and could not initiate selection of a new arbitrator.

  Now in such cases, when the arbitrator fails or loses contact during arbitration, the requesting node immediately fails rather than waiting to time out. (Bug #20538179)

- `DROP DATABASE` failed to remove the database when the database directory contained a `.ndb` file which had no corresponding table in `NDB`. Now, when executing `DROP DATABASE`, `NDB` performs an check specifically for leftover `.ndb` files, and deletes any that it finds. (Bug #20480035)

  References: See also: Bug #44529.

- When performing a restart, it was sometimes possible to find a log end marker which had been written by a previous restart, and that should have been invalidated. Now when searching for the last page to invalidate, the same search algorithm is used as when searching for the last page of the log to read. (Bug #76207, Bug #20665205)

- During a node restart, if there was no global checkpoint completed between the `START_LCP_REQ` for a local checkpoint and its `LCP_COMPLETE_REP` it was possible for a comparison of the LCP ID sent in the `LCP_COMPLETE_REP` signal with the internal value `SYSFILE->latestLCP_ID` to fail. (Bug #76113, Bug #20631645)

- When sending `LCP_FRAG_ORD` signals as part of master takeover, it is possible that the master not is not synchronized with complete accuracy in real time, so that some signals must be dropped. During this time, the master can send a `LCP_FRAG_ORD` signal with its `lastFragmentFlag` set even after the local checkpoint has been completed. This enhancement causes this flag to persist until the statrt of the next local checkpoint, which causes these signals to be dropped as well.

This change affects `ndbd` only; the issue described did not occur with `ndbmtd`. (Bug #75964, Bug #20567730)

- When reading and copying transporter short signal data, it was possible for the data to be copied back to the same signal with overlapping memory. (Bug #75930, Bug #20553247)

- NDB node takeover code made the assumption that there would be only one takeover record when starting a takeover, based on the further assumption that the master node could never perform copying of fragments. However, this is not the case in a system restart, where a master node can have stale data and so need to perform such copying to bring itself up to date. (Bug #75919, Bug #20546899)

- **NDB Cluster APIs:** A scan operation, whether it is a single table scan or a query scan used by a pushed join, stores the result set in a buffer. This maximum size of this buffer is calculated and preallocated before the scan operation is started. This buffer may consume a considerable amount of memory; in some cases we observed a 2 GB buffer footprint in tests that executed 100 parallel scans with 2 single-threaded (`ndbd`) data nodes. This memory consumption was found to scale linearly with additional fragments.

  A number of root causes, listed here, were discovered that led to this problem:

  - Result rows were unpacked to full `NdbRecord` format before they were stored in the buffer. If only some but not all columns of a table were selected, the buffer contained empty space (essentially wasted).

  - Due to the buffer format being unpacked, `VARCHAR` and `VARBINARY` columns always had to be allocated for the maximum size defined for such columns.

  - `BatchByteSize` and `MaxScanBatchSize` values were not taken into consideration as a limiting factor when calculating the maximum buffer size.

  These issues became more evident in NDB 7.2 and later MySQL Cluster release series. This was due to the fact buffer size is scaled by `BatchSize`, and that the default value for this parameter was increased fourfold (from 64 to 256) beginning with MySQL Cluster NDB 7.2.1.

  This fix causes result rows to be buffered using the packed format instead of the unpacked format; a buffered scan result row is now not unpacked until it becomes the current row. In addition, `BatchByteSize` and `MaxScanBatchSize` are now used as limiting factors when calculating the required buffer size.

  Also as part of this fix, refactoring has been done to separate handling of buffered (packed) from handling of unbuffered result sets, and to remove code that had been unused since NDB 7.0 or earlier. The `NdbRecord` class declaration has also been cleaned up by removing a number of unused or redundant member variables. (Bug #73781, Bug #75599, Bug #19631350, Bug #20408733)

Changes in MySQL NDB Cluster 7.4.4 (5.6.23-ndb-7.4.4) (2015-02-26, General Availability)

MySQL Cluster NDB 7.4.4 is the first GA release of MySQL Cluster NDB 7.4, based on MySQL Server 5.6 and including new features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous MySQL Cluster releases.

Obtaining MySQL Cluster NDB 7.4.   MySQL Cluster NDB 7.4 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.4, see What is New in NDB Cluster 7.4.

This release also incorporates all bugfixes and changes made in previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.23 (see Changes in MySQL 5.6.23 (2015-02-02, General Availability)).

Bugs Fixed

- When upgrading a MySQL Cluster from NDB 7.3 to NDB 7.4, the first data node started with the NDB 7.4 data node binary caused the master node (still running NDB 7.3) to fail with Error 2301, then itself failed during Start Phase 5. (Bug #20608889)

- A memory leak in NDB event buffer allocation caused an event to be leaked for each epoch. (Due to the fact that an SQL node uses 3 event buffers, each SQL node leaked 3 events per epoch.) This meant that a MySQL Cluster `mysqld` leaked an amount of memory that was inversely proportional to the size of `TimeBetweenEpochs`—that is, the smaller the value for this parameter, the greater the amount of memory leaked per unit of time. (Bug #20539452)

- The values of the `Ndb_last_commit_epoch_server` and `Ndb_last_commit_epoch_session` status variables were incorrectly reported on some platforms. To correct this problem, these values are now stored internally as `long long`, rather than `long`. (Bug #20372169)

- When restoring a MySQL Cluster from backup, nodes that failed and were restarted during restoration of another node became unresponsive, which subsequently caused `ndb_restore` to fail and exit. (Bug #20069066)

- When a data node fails or is being restarted, the remaining nodes in the same nodegroup resend to subscribers any data which they determine has not already been sent by the failed node. Normally, when a data node (actually, the `SUMA` kernel block) has sent all data belonging to an epoch for which it is responsible, it sends a `SUB_GCP_COMPLETE_REP` signal, together with a count, to all subscribers, each of which responds with a `SUB_GCP_COMPLETE_ACK`. When `SUMA` receives this acknowledgment from all subscribers, it reports this to the other nodes in the same nodegroup so that they know that there is no need to resend this data in case of a subsequent node failure. If a node failed before all subscribers sent this acknowledgement but before all the other nodes in the same nodegroup received it from the failing node, data for some epochs could be sent (and reported as complete) twice, which could lead to an unplanned shutdown.

  The fix for this issue adds to the count reported by `SUB_GCP_COMPLETE_ACK` a list of identifiers which the receiver can use to keep track of which buckets are completed and to ignore any duplicate reported for an already completed bucket. (Bug #17579998)

- The `ndbinfo.restart_info` table did not contain a new row as expected following a node restart. (Bug #75825, Bug #20504971)

- The output format of `SHOW CREATE TABLE` for an `NDB` table containing foreign key constraints did not match that for the equivalent `InnoDB` table, which could lead to issues with some third-party applications. (Bug #75515, Bug #20364309)

- An `ALTER TABLE` statement containing comments and a partitioning option against an `NDB` table caused the SQL node on which it was executed to fail. (Bug #74022, Bug #19667566)

- **NDB Cluster APIs:** When a transaction is started from a cluster connection, `Table` and `Index` schema objects may be passed to this transaction for use. If these schema objects have been acquired from a different connection (`Ndb_cluster_connection` object), they can be deleted at any point by the deletion or disconnection of the owning connection. This can leave a connection with invalid schema objects, which causes an NDB API application to fail when these are dereferenced.

  To avoid this problem, if your application uses multiple connections, you can now set a check to detect sharing of schema objects between connections when passing a schema object to a transaction, using the `NdbTransaction::setSchemaObjectOwnerChecks()` method added in this release. When this check is enabled, the schema objects having the same names are acquired

from the connection and compared to the schema objects passed to the transaction. Failure to match causes the application to fail with an error. (Bug #19785977)

- **NDB Cluster APIs:** The increase in the default number of hashmap buckets (`DefaultHashMapSize` API node configuration parameter) from 240 to 3480 in MySQL Cluster NDB 7.2.11 increased the size of the internal `DictHashMapInfo::HashMap` type considerably. This type was allocated on the stack in some `getTable()` calls which could lead to stack overflow issues for NDB API users.

  To avoid this problem, the hashmap is now dynamically allocated from the heap. (Bug #19306793)

Changes in MySQL NDB Cluster 7.4.3 (5.6.22-ndb-7.4.3) (2015-01-21, Release Candidate)

MySQL Cluster NDB 7.4.3 is a new release of MySQL Cluster, based on MySQL Server 5.6 and including features under development for version 7.4 of the `NDB` storage engine, as well as fixing a number of recently discovered bugs in previous MySQL Cluster releases.

Obtaining MySQL Cluster NDB 7.4.   MySQL Cluster NDB 7.4 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.4, see What is New in NDB Cluster 7.4.

This release also incorporates all bugfixes and changes made in previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.22 (see Changes in MySQL 5.6.22 (2014-12-01, General Availability)).

- Functionality Added or Changed

- Bugs Fixed

Functionality Added or Changed

- **Important Change; NDB Cluster APIs:** This release introduces an epoch-driven Event API for the NDB API that supercedes the earlier GCI-based model. The new version of this API also simplifies error detection and handling, and monitoring of event buffer memory usage has been been improved.

  New event handling methods for `Ndb` and `NdbEventOperation` added by this change include `NdbEventOperation::getEventType2()`, `pollEvents2()`, `nextEvent2()`, `getHighestQueuedEpoch()`, `getNextEventOpInEpoch2()`, `getEpoch()`, `isEmptyEpoch()`, and `isErrorEpoch`. The `pollEvents()`, `nextEvent()`, `getLatestGCI()`, `getGCIEventOperations()`, `isConsistent()`, `isConsistentGCI()`, `getEventType()`, `getGCI()`, `getLatestGCI()`, `isOverrun()`, `hasError()`, and `clearError()` methods are deprecated beginning with the same release.

  Some (but not all) of the new methods act as replacements replacements for deprecated methods; not all of the deprecated methods map to new ones. The Event Class, provides information as to which old methods correspond to new ones.

  Error handling using the new API is no longer handled using dedicated `hasError()` and `clearError()` methods, which are now deprecated as previously noted. To support this change, `TableEvent` now supports the values `TE_EMPTY` (empty epoch), `TE_INCONSISTENT` (inconsistent epoch), and `TE_OUT_OF_MEMORY` (insufficient event buffer memory).

  Event buffer memory management has also been improved with the introduction of the `get_eventbuffer_free_percent()`, `set_eventbuffer_free_percent()`, and `get_event_buffer_memory_usage()` methods, as well as a new NDB API error `Free percent out of range` (error code 4123). Memory buffer usage can now be represented in applications using the `EventBufferMemoryUsage` data structure, and checked from MySQL client applications by reading the `ndb_eventbuffer_free_percent` system variable.

For more information, see the detailed descriptions for the `Ndb` and `NdbEventOperation` methods listed. See also Event::TableEvent, as well as The EventBufferMemoryUsage Structure.

- Additional logging is now performed of internal states occurring during system restarts such as waiting for node ID allocation and master takeover of global and local checkpoints. (Bug #74316, Bug #19795029)

- Added the `operations_per_fragment` table to the `ndbinfo` information database. Using this table, you can now obtain counts of operations performed on a given fragment (or fragment replica). Such operations include reads, writes, updates, and deletes, scan and index operations performed while executing them, and operations refused, as well as information relating to rows scanned on and returned from a given fragment replica. This table also provides information about interpreted programs used as attribute values, and values returned by them.

- Added the `MaxParallelCopyInstances` data node configuration parameter. In cases where the parallelism used during restart copy phase (normally the number of LDMs up to a maximum of 16) is excessive and leads to system overload, this parameter can be used to override the default behavior by reducing the degree of parallelism employed.

- **NDB Cluster APIs:** Two new example programs, demonstrating reads and writes of `CHAR`, `VARCHAR`, and `VARBINARY` column values, have been added to `storage/ndb/ndbapi-examples` in the MySQL Cluster source tree. For more information about these programs, including source code listings, see NDB API Simple Array Example, and NDB API Simple Array Example Using Adapter.

Bugs Fixed

- The global checkpoint commit and save protocols can be delayed by various causes, including slow disk I/O. The `DIH` master node monitors the progress of both of these protocols, and can enforce a maximum lag time during which the protocols are stalled by killing the node responsible for the lag when it reaches this maximum. This `DIH` master GCP monitor mechanism did not perform its task more than once per master node; that is, it failed to continue monitoring after detecting and handling a GCP stop. (Bug #20128256)

  References: See also: Bug #19858151, Bug #20069617, Bug #20062754.

- When running `mysql_upgrade` on a MySQL Cluster SQL node, the expected drop of the `performance_schema` database on this node was instead performed on all SQL nodes connected to the cluster. (Bug #20032861)

- The warning shown when an `ALTER TABLE ALGORITHM=INPLACE ... ADD COLUMN` statement automatically changes a column's `COLUMN_FORMAT` from `FIXED` to `DYNAMIC` now includes the name of the column whose format was changed. (Bug #20009152, Bug #74795)

- The local checkpoint scan fragment watchdog and the global checkpoint monitor can each exclude a node when it is too slow when participating in their respective protocols. This exclusion was implemented by simply asking the failing node to shut down, which in case this was delayed (for whatever reason) could prolong the duration of the GCP or LCP stall for other, unaffected nodes.

  To minimize this time, an isolation mechanism has been added to both protocols whereby any other live nodes forcibly disconnect the failing node after a predetermined amount of time. This allows the failing node the opportunity to shut down gracefully (after logging debugging and other information) if possible, but limits the time that other nodes must wait for this to occur. Now, once the remaining live nodes have processed the disconnection of any failing nodes, they can commence failure handling and restart the related protocol or protocol, even if the failed node takes an excessively long time to shut down. (Bug #19858151)

  References: See also: Bug #20128256, Bug #20069617, Bug #20062754.

- The matrix of values used for thread configuration when applying the setting of the `MaxNoOfExecutionThreads` configuration parameter has been improved to align with support for

greater numbers of LDM threads. See Multi-Threading Configuration Parameters (ndbmtd), for more information about the changes. (Bug #75220, Bug #20215689)

- When a new node failed after connecting to the president but not to any other live node, then reconnected and started again, a live node that did not see the original connection retained old state information. This caused the live node to send redundant signals to the president, causing it to fail. (Bug #75218, Bug #20215395)

- In the `NDB` kernel, it was possible for a `TransporterFacade` object to reset a buffer while the data contained by the buffer was being sent, which could lead to a race condition. (Bug #75041, Bug #20112981)

- `mysql_upgrade` failed to drop and recreate the `ndbinfo` database and its tables as expected. (Bug #74863, Bug #20031425)

- Due to a lack of memory barriers, MySQL Cluster programs such as `ndbmtd` did not compile on `POWER` platforms. (Bug #74782, Bug #20007248)

- In spite of the presence of a number of protection mechanisms against overloading signal buffers, it was still in some cases possible to do so. This fix adds block-level support in the `NDB` kernel (in `SimulatedBlock`) to make signal buffer overload protection more reliable than when implementing such protection on a case-by-case basis. (Bug #74639, Bug #19928269)

- Copying of metadata during local checkpoints caused node restart times to be highly variable which could make it difficult to diagnose problems with restarts. The fix for this issue introduces signals (including `PAUSE_LCP_IDLE`, `PAUSE_LCP_REQUESTED`, and `PAUSE_NOT_IN_LCP_COPY_META_DATA`) to pause LCP execution and flush LCP reports, making it possible to block LCP reporting at times when LCPs during restarts become stalled in this fashion. (Bug #74594, Bug #19898269)

- When a data node was restarted from its angel process (that is, following a node failure), it could be allocated a new node ID before failure handling was actually completed for the failed node. (Bug #74564, Bug #19891507)

- In `NDB` version 7.4, node failure handling can require completing checkpoints on up to 64 fragments. (This checkpointing is performed by the `DBLQH` kernel block.) The requirement for master takeover to wait for completion of all such checkpoints led in such cases to excessive length of time for completion.

  To address these issues, the `DBLQH` kernel block can now report that it is ready for master takeover before it has completed any ongoing fragment checkpoints, and can continue processing these while the system completes the master takeover. (Bug #74320, Bug #19795217)

- Local checkpoints were sometimes started earlier than necessary during node restarts, while the node was still waiting for copying of the data distribution and data dictionary to complete. (Bug #74319, Bug #19795152)

- The check to determine when a node was restarting and so know when to accelerate local checkpoints sometimes reported a false positive. (Bug #74318, Bug #19795108)

- Values in different columns of the `ndbinfo` tables `disk_write_speed_aggregate` and `disk_write_speed_aggregate_node` were reported using differing multiples of bytes. Now all of these columns display values in bytes.

  In addition, this fix corrects an error made when calculating the standard deviations used in the `std_dev_backup_lcp_speed_last_10sec`, `std_dev_redo_speed_last_10sec`, `std_dev_backup_lcp_speed_last_60sec`, and `std_dev_redo_speed_last_60sec` columns of the `ndbinfo.disk_write_speed_aggregate` table. (Bug #74317, Bug #19795072)

- Recursion in the internal method `Dblqh::finishScanrec()` led to an attempt to create two list iterators with the same head. This regression was introduced during work done to optimize scans for version 7.4 of the `NDB` storage engine. (Bug #73667, Bug #19480197)

- Transporter send buffers were not updated properly following a failed send. (Bug #45043, Bug #20113145)

- **NDB Disk Data:** An update on many rows of a large Disk Data table could in some rare cases lead to node failure. In the event that such problems are observed with very large transactions on Disk Data tables you can now increase the number of page entries allocated for disk page buffer memory by raising the value of the `DiskPageBufferEntries` data node configuration parameter added in this release. (Bug #19958804)

- **NDB Disk Data:** In some cases, during `DICT` master takeover, the new master could crash while attempting to roll forward an ongoing schema transaction. (Bug #19875663, Bug #74510)

- **NDB Cluster APIs:** It was possible to delete an `Ndb_cluster_connection` object while there remained instances of `Ndb` using references to it. Now the `Ndb_cluster_connection` destructor waits for all related `Ndb` objects to be released before completing. (Bug #19999242)

  References: See also: Bug #19846392.

- **MySQL NDB ClusterJ:** ClusterJ reported a segmentation violation when an application closed a session factory while some sessions were still active. This was because MySQL Cluster allowed an `Ndb_cluster_connection` object be to deleted while some `Ndb` instances were still active, which might result in the usage of null pointers by ClusterJ. This fix stops that happening by preventing ClusterJ from closing a session factory when any of its sessions are still active. (Bug #19846392)

  References: See also: Bug #19999242.

Changes in MySQL NDB Cluster 7.4.2 (5.6.21-ndb-7.4.2) (2014-11-05, Development Milestone)

MySQL Cluster NDB 7.4.2 is a new release of MySQL Cluster, based on MySQL Server 5.6 and including features under development for version 7.4 of the `NDB` storage engine, as well as fixing a number of recently discovered bugs in previous MySQL Cluster releases.

Obtaining MySQL Cluster NDB 7.4.   MySQL Cluster NDB 7.4 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.4, see What is New in NDB Cluster 7.4.

This release also incorporates all bugfixes and changes made in previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.21 (see Changes in MySQL 5.6.21 (2014-09-23, General Availability)).

- Functionality Added or Changed

- Bugs Fixed

Functionality Added or Changed

- Added the `restart_info` table to the `ndbinfo` information database to provide current status and timing information relating to node and system restarts. By querying this table, you can observe the progress of restarts in real time. (Bug #19795152)

- After adding new data nodes to the configuration file of a MySQL Cluster having many API nodes, but prior to starting any of the data node processes, API nodes tried to connect to these "missing" data nodes several times per second, placing extra loads on management nodes and the network. To reduce unnecessary traffic caused in this way, it is now possible to control the amount of time that an API node waits between attempts to connect to data nodes which fail to respond; this is implemented in two new API node configuration parameters `StartConnectBackoffMaxTime` and `ConnectBackoffMaxTime`.

  Time elapsed during node connection attempts is not taken into account when applying these parameters, both of which are given in milliseconds with approximately 100 ms resolution. As

long as the API node is not connected to any data nodes as described previously, the value of the `StartConnectBackoffMaxTime` parameter is applied; otherwise, `ConnectBackoffMaxTime` is used.

In a MySQL Cluster with many unstarted data nodes, the values of these parameters can be raised to circumvent connection attempts to data nodes which have not yet begun to function in the cluster, as well as moderate high traffic to management nodes.

For more information about the behavior of these parameters, see Defining SQL and Other API Nodes in an NDB Cluster. (Bug #17257842)

Bugs Fixed

- When performing a batched update, where one or more successful write operations from the start of the batch were followed by write operations which failed without being aborted (due to the `AbortOption` being set to `AO_IgnoreError`), the failure handling for these by the transaction coordinator leaked `CommitAckMarker` resources. (Bug #19875710)

  References: This issue is a regression of: Bug #19451060, Bug #73339.

- Online downgrades to MySQL Cluster NDB 7.3 failed when a MySQL Cluster NDB 7.4 master attempted to request a local checkpoint with 32 fragments from a data node already running NDB 7.3, which supports only 2 fragments for LCPs. Now in such cases, the NDB 7.4 master determines how many fragments the data node can handle before making the request. (Bug #19600834)

- The fix for a previous issue with the handling of multiple node failures required determining the number of TC instances the failed node was running, then taking them over. The mechanism to determine this number sometimes provided an invalid result which caused the number of TC instances in the failed node to be set to an excessively high value. This in turn caused redundant takeover attempts, which wasted time and had a negative impact on the processing of other node failures and of global checkpoints. (Bug #19193927)

  References: This issue is a regression of: Bug #18069334.

- The server side of an NDB transporter disconnected an incoming client connection very quickly during the handshake phase if the node at the server end was not yet ready to receive connections from the other node. This led to problems when the client immediately attempted once again to connect to the server socket, only to be disconnected again, and so on in a repeating loop, until it suceeded. Since each client connection attempt left behind a socket in `TIME_WAIT`, the number of sockets in `TIME_WAIT` increased rapidly, leading in turn to problems with the node on the server side of the transporter.

  Further analysis of the problem and code showed that the root of the problem lay in the handshake portion of the transporter connection protocol. To keep the issue described previously from occurring, the node at the server end now sends back a `WAIT` message instead of disconnecting the socket when the node is not yet ready to accept a handshake. This means that the client end should no longer need to create a new socket for the next retry, but can instead begin immediately with a new handshake hello message. (Bug #17257842)

- Corrupted messages to data nodes sometimes went undetected, causing a bad signal to be delivered to a block which aborted the data node. This failure in combination with disconnecting nodes could in turn cause the entire cluster to shut down.

  To keep this from happening, additional checks are now made when unpacking signals received over TCP, including checks for byte order, compression flag (which must not be used), and the length of the next message in the receive buffer (if there is one).

  Whenever two consecutive unpacked messages fail the checks just described, the current message is assumed to be corrupted. In this case, the transporter is marked as having bad data and no more unpacking of messages occurs until the transporter is reconnected. In addition, an entry is written to

the cluster log containing the error as well as a hex dump of the corrupted message. (Bug #73843, Bug #19582925)

- During restore operations, an attribute's maximum length was used when reading variable-length attributes from the receive buffer instead of the attribute's actual length. (Bug #73312, Bug #19236945)

- **NDB Replication:** The fix for Bug #18770469 in the MySQL Server made changes in the transactional behavior of the temporary conversion tables used when replicating between tables with different schemas. These changes as implemented are not compatible with `NDB`, and thus the fix for this bug has been reverted in MySQL Cluster. (Bug #19692387)

  References: See also: Bug #19704825. Reverted patches: Bug #18770469.

Changes in MySQL NDB Cluster 7.4.1 (5.6.20-ndb-7.4.1) (2014-09-25, Development Milestone)

MySQL Cluster NDB 7.4.1 is a new Developer Milestone release of MySQL Cluster, based on MySQL Server 5.6 and previewing new features under development for version 7.4 of the `NDB` storage engine.

Obtaining MySQL Cluster NDB 7.4. MySQL Cluster NDB 7.4 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.4, see What is New in NDB Cluster 7.4.

This release also incorporates all bugfixes and changes made in previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.20 (see Changes in MySQL 5.6.20 (2014-07-31, General Availability)).

- Conflict Resolution Exceptions Table Extensions

- Node Restart Performance and Reporting Enhancements

- Dynamic Primary/Secondary Role Determination

- Improved Scan and SQL Processing

- Per-Fragment Memory Reporting

- Bugs Fixed

Conflict Resolution Exceptions Table Extensions

- **NDB Replication:** A number of changes and improvements have been made to exceptions tables for MySQL Cluster Replication conflict detection and resolution. A reserved column name namespace is now employed for metacolumns, which allows the recording of an arbitrary subset of main table columns that are not part of the table's primary key. The names of all metacolumns in the exception table should now be prefixed with `NDB$`.

  It is no longer necessary to record the complete primary key. Matching of main table columns to exceptions table columns is now performed solely on the basis of name and type. In addition, you can now record in the exceptions table the values of columns which not part of the main table's primary key.

  Predefined optional columns can now be employed in conflict exceptions tables to obtain information about a conflict's type, cause, and originating transaction.

  Read tracking—that is, detecting conflicts between reads of a given row in one cluster and updates or deletes of the same row in another cluster—is now supported. This requires exclusive read locks obtained by setting `ndb_log_exclusive_reads` equal to 1 on the slave cluster. All rows read by

a conflicting read are logged in the exceptions table. For more information and examples, see Read conflict detection and resolution.

Existing exceptions tables continue to be supported. For additional information, see Conflict resolution exceptions table.

Node Restart Performance and Reporting Enhancements

- **Performance:** A number of performance and other improvements have been made with regard to node starts and restarts. The following list contains a brief description of each of these changes:

  - Before memory allocated on startup can be used, it must be touched, causing the operating system to allocate the actual physical memory needed. The process of touching each page of memory that was allocated has now been multithreaded, with touch times on the order of 3 times shorter than with a single thread when performed by 16 threads.

  - When performing a node or system restart, it is necessary to restore local checkpoints for the fragments. This process previously used delayed signals at a point which was found to be critical to performance; these have now been replaced with normal (undelayed) signals, which should shorten significantly the time required to back up a MySQL Cluster or to restore it from backup.

  - Previously, there could be at most 2 LDM instances active with local checkpoints at any given time. Now, up to 16 LDMs can be used for performing this task, which increases utilization of available CPU power, and can speed up LCPs by a factor of 10, which in turn can greatly improve restart times.

    Better reporting of disk writes and increased control over these also make up a large part of this work. New `ndbinfo` tables `disk_write_speed_base`, `disk_write_speed_aggregate`, and `disk_write_speed_aggregate_node` provide information about the speed of disk writes for each LDM thread that is in use. The `DiskCheckpointSpeed` and `DiskCheckpointSpeedInRestart` configuration parameters have been deprecated, and are subject to removal in a future MySQL Cluster release. This release adds the data node configuration parameters `MinDiskWriteSpeed`, `MaxDiskWriteSpeed`, `MaxDiskWriteSpeedOtherNodeRestart`, and `MaxDiskWriteSpeedOwnRestart` to control write speeds for LCPs and backups when the present node, another node, or no node is currently restarting.

    For more information, see the descriptions of the `ndbinfo` tables and MySQL Cluster configuration parameters named previously.

  - Reporting of MySQL Cluster start phases has been improved, with more frequent printouts. New and better information about the start phases and their implementation has also been provided in the sources and documentation. See Summary of NDB Cluster Start Phases.

Dynamic Primary/Secondary Role Determination

- **NDB Replication:** When using conflict detection and resolution with a circular or "active-active" MySQL Cluster Replication setup, it is now possible to set the roles of primary and secondary cluster explicitly and dynamically by setting the `ndb_slave_conflict_role` server system variable introduced in this release. This variable can take any one of the values `PRIMARY`, `SECONDARY`, `PASS`, or `NULL` (the default). (`PASS` enables a passthrough state in which the effects of any conflict resolution function are ignored.) This can be useful when it is necessary to fail over from the MySQL Cluster acting as the primary.

  The slave SQL thread must be stopped when the value of this variable is changed. In addition, it is not possible to change it directly between `PASS` and either of `PRIMARY` or `SECONDARY`.

  For more information, see the description of `ndb_slave_conflict_role` as well as NDB Cluster Replication Conflict Resolution.

Improved Scan and SQL Processing

- **Performance:** Several internal methods relating to the `NDB` receive thread have been optimized to make `mysqld` more efficient in processing SQL applications with the `NDB` storage engine. In particular, this work improves the performance of the `NdbReceiver::execTRANSID_AI()` method, which is commonly used to receive a record from the data nodes as part of a scan operation. (Since the receiver thread sometimes has to process millions of received records per second, it is critical that this method does not perform unnecessary work, or tie up resources that are not strictly needed.) The associated internal functions `receive_ndb_packed_record()` and `handleReceivedSignal()` methods have also been improved, and made more efficient.

Per-Fragment Memory Reporting

- Information about memory usage by individual fragments can now be obtained from the `memory_per_fragment` view added in this release to the `ndbinfo` information database. This information includes pages having fixed, and variable element size, rows, fixed element free slots, variable element free bytes, and hash index memory usage. For information, see The ndbinfo memory_per_fragment Table.

Bugs Fixed

- In some cases, transporter receive buffers were reset by one thread while being read by another. This happened when a race condition occurred between a thread receiving data and another thread initiating disconnect of the transporter (disconnection clears this buffer). Concurrency logic has now been implemented to keep this race from taking place. (Bug #19552283, Bug #73790)

- When a new data node started, API nodes were allowed to attempt to register themselves with the data node for executing transactions before the data node was ready. This forced the API node to wait an extra heartbeat interval before trying again.

  To address this issue, a number of **HA_ERR_NO_CONNECTION** errors (Error 4009) that could be issued during this time have been changed to `Cluster temporarily unavailable` errors (Error 4035), which should allow API nodes to use new data nodes more quickly than before. As part of this fix, some errors which were incorrectly categorised have been moved into the correct categories, and some errors which are no longer used have been removed. (Bug #19524096, Bug #73758)

- Executing `ALTER TABLE ... REORGANIZE PARTITION` after increasing the number of data nodes in the cluster from 4 to 16 led to a crash of the data nodes. This issue was shown to be a regression caused by previous fix which added a new dump handler using a dump code that was already in use (7019), which caused the command to execute two different handlers with different semantics. The new handler was assigned a new `DUMP` code (7024). (Bug #18550318)

  References: This issue is a regression of: Bug #14220269.

- When certain queries generated signals having more than 18 data words prior to a node failure, such signals were not written correctly in the trace file. (Bug #18419554)

- Failure of multiple nodes while using `ndbmtd` with multiple TC threads was not handled gracefully under a moderate amount of traffic, which could in some cases lead to an unplanned shutdown of the cluster. (Bug #18069334)

- For multithreaded data nodes, some threads do communicate often, with the result that very old signals can remain at the top of the signal buffers. When performing a thread trace, the signal dumper calculated the latest signal ID from what it found in the signal buffers, which meant that these old signals could be erroneously counted as the newest ones. Now the signal ID counter is kept as part of the thread state, and it is this value that is used when dumping signals for trace files. (Bug #73842, Bug #19582807)

- **NDB Cluster APIs:** When an NDB API client application received a signal with an invalid block or signal number, `NDB` provided only a very brief error message that did not accurately convey the nature of the problem. Now in such cases, appropriate printouts are provided when a bad signal or message is detected. In addition, the message length is now checked to make certain that it matches the size of the embedded signal. (Bug #18426180)

Release Series Changelogs: MySQL Cluster NDB 7.4

This section contains unified changelog information for the MySQL Cluster NDB 7.4 release series.

For changelogs covering individual MySQL Cluster NDB 7.4 releases, see NDB Cluster Release Notes.

For general information about features added in MySQL Cluster NDB 7.4, see What is New in NDB Cluster 7.4.

For an overview of features added in MySQL 5.6 that are not specific to MySQL Cluster, see What Is New in MySQL 5.6. For a complete list of all bugfixes and feature changes made in MySQL 5.6 that are not specific to MySQL Cluster, see the MySQL 5.6 Release Notes.

Changes in the MySQL Cluster NDB 7.4 Series

This section contains unified change history highlights for all MySQL Cluster releases based on version 7.4 of the `NDB` storage engine through MySQL Cluster NDB 7.4.14. Included are all changelog entries in the categories *MySQL Cluster*, *Disk Data*, and *Cluster API*.

For an overview of features that were added in MySQL Cluster NDB 7.4, see What is New in NDB Cluster 7.4.

- Changes in MySQL NDB Cluster 7.4.12 (5.6.31-ndb-7.4.12)

- Changes in MySQL NDB Cluster 7.4.11 (5.6.29-ndb-7.4.11)

- Changes in MySQL NDB Cluster 7.4.10 (5.6.28-ndb-7.4.10)

- Changes in MySQL NDB Cluster 7.4.9 (5.6.28-ndb-7.4.9)

- Changes in MySQL NDB Cluster 7.4.8 (5.6.27-ndb-7.4.8)

- Changes in MySQL NDB Cluster 7.4.7 (5.6.25-ndb-7.4.7)

- Changes in MySQL NDB Cluster 7.4.6 (5.6.24-ndb-7.4.6)

- Changes in MySQL NDB Cluster 7.4.5 (5.6.23-ndb-7.4.5)

- Changes in MySQL NDB Cluster 7.4.4 (5.6.23-ndb-7.4.4)

- Changes in MySQL NDB Cluster 7.4.3 (5.6.22-ndb-7.4.3)

- Changes in MySQL NDB Cluster 7.4.2 (5.6.21-ndb-7.4.2)

- Changes in MySQL NDB Cluster 7.4.1 (5.6.20-ndb-7.4.1)

Changes in MySQL NDB Cluster 7.4.12 (5.6.31-ndb-7.4.12)

Bugs Fixed

- **Incompatible Change:** When the data nodes are only partially connected to the API nodes, a node used for a pushdown join may get its request from a transaction coordinator on a different node, without (yet) being connected to the API node itself. In such cases, the `NodeInfo` object for the requesting API node contained no valid info about the software version of the API node, which caused the `DBSPJ` block to assume (incorrectly) when aborting to assume that the API node used `NDB` version 7.2.4 or earlier, requiring the use of a backward compatability mode to be used during query abort which sent a node failure error instead of the real error causing the abort.

  Now, whenever this situation occurs, it is assumed that, if the `NDB` software version is not yet available, the API node version is greater than 7.2.4. (Bug #23049170)

- Although arguments to the `DUMP` command are 32-bit integers, `ndb_mgmd` used a buffer of only 10 bytes when processing them. (Bug #23708039)

- During shutdown, the `mysqld` process could sometimes hang after logging `NDB Util: Stop` ... `NDB Util: Wakeup`. (Bug #23343739)

  References: See also: Bug #21098142.

- During an online upgrade from a MySQL Cluster NDB 7.3 release to an NDB 7.4 (or later) release, the failures of several data nodes running the lower version during local checkpoints (LCPs), and just prior to upgrading these nodes, led to additional node failures following the upgrade. This was due to lingering elements of the `EMPTY_LCP` protocol initiated by the older nodes as part of an LCP-plus-restart sequence, and which is no longer used in NDB 7.4 and later due to LCP optimizations implemented in those versions. (Bug #23129433)

- Reserved send buffer for the loopback transporter, introduced in MySQL Cluster NDB 7.4.8 and used by API and management nodes for administrative signals, was calculated incorrectly. (Bug #23093656, Bug #22016081)

  References: This issue is a regression of: Bug #21664515.

- During a node restart, re-creation of internal triggers used for verifying the referential integrity of foreign keys was not reliable, because it was possible that not all distributed TC and LDM instances agreed on all trigger identities. To fix this problem, an extra step is added to the node restart sequence, during which the trigger identities are determined by querying the current master node. (Bug #23068914)

  References: See also: Bug #23221573.

- Following the forced shutdown of one of the 2 data nodes in a cluster where `NoOfReplicas=2`, the other data node shut down as well, due to arbitration failure. (Bug #23006431)

- The `ndbinfo.tc_time_track_stats` table uses histogram buckets to give a sense of the distribution of latencies. The sizes of these buckets were also reported as `HISTOGRAM BOUNDARY INFO` messages during data node startup; this printout was redundant and so has been removed. (Bug #22819868)

- A failure occurred in `DBTUP` in debug builds when variable-sized pages for a fragment totalled more than 4 GB. (Bug #21313546)

- `mysqld` did not shut down cleanly when executing `ndb_index_stat`. (Bug #21098142)

  References: See also: Bug #23343739.

- `DBDICT` and `GETTABINFOREQ` queue debugging were enhanced as follows:

  - Monitoring by a data node of the progress of `GETTABINFOREQ` signals can be enabled by setting `DictTrace` >= 2.

  - Added the `ApiVerbose` configuration parameter, which enables NDB API debug logging for an API node where it is set greater than or equal to 2.

  - Added `DUMP` code 1229 which shows the current state of the `GETTABINFOREQ` queue. (See DUMP 1229.)

  See also The DBDICT Block. (Bug #20368450)

  References: See also: Bug #20368354.

- **NDB Cluster APIs:** Deletion of Ndb objects used a dispoportionately high amount of CPU. (Bug #22986823)

Changes in MySQL NDB Cluster 7.4.11 (5.6.29-ndb-7.4.11)

Functionality Added or Changed

- **NDB Cluster APIs:** Added the `Ndb::setEventBufferQueueEmptyEpoch()` method, which makes it possible to enable queuing of empty events (event type `TE_EMPTY`). (Bug #22157845)

Bugs Fixed

- **Important Change:** The minimum value for the `BackupDataBufferSize` data node configuration parameter has been lowered from 2 MB to 512 KB. The default and maximum values for this parameter remain unchanged. (Bug #22749509)

- **Microsoft Windows:** Performing `ANALYZE TABLE` on a table having one or more indexes caused `ndbmtd` to fail with an `InvalidAttrInfo` error due to signal corruption. This issue occurred consistently on Windows, but could also be encountered on other platforms. (Bug #77716, Bug #21441297)

- During node failure handling, the request structure used to drive the cleanup operation was not maintained correctly when the request was executed. This led to inconsistencies that were harmless during normal operation, but these could lead to assertion failures during node failure handling, with subsequent failure of additional nodes. (Bug #22643129)

- The previous fix for a lack of mutex protection for the internal `TransporterFacade::deliver_signal()` function was found to be incomplete in some cases. (Bug #22615274)

  References: This issue is a regression of: Bug #77225, Bug #21185585.

- Compilation of MySQL with Visual Studio 2015 failed in `ConfigInfo.cpp`, due to a change in Visual Studio's handling of spaces and concatenation. (Bug #22558836, Bug #80024)

- When setup of the binary log as an atomic operation on one SQL node failed, this could trigger a state in other SQL nodes in which they appeared to detect the SQL node participating in schema change distribution, whereas it had not yet completed binary log setup. This could in turn cause a deadlock on the global metadata lock when the SQL node still retrying binary log setup needed this lock, while another mysqld had taken the lock for itself as part of a schema change operation. In such cases, the second SQL node waited for the first one to act on its schema distribution changes, which it was not yet able to do. (Bug #22494024)

- Duplicate key errors could occur when `ndb_restore` was run on a backup containing a unique index. This was due to the fact that, during restoration of data, the database can pass through one or more inconsistent states prior to completion, such an inconsistent state possibly having duplicate values for a column which has a unique index. (If the restoration of data is preceded by a run with `--disable-indexes` and followed by one with `--rebuild-indexes`, these errors are avoided.)

  Added a check for unique indexes in the backup which is performed only when restoring data, and which does not process tables that have explicitly been excluded. For each unique index found, a warning is now printed. (Bug #22329365)

- Restoration of metadata with `ndb_restore -m` occasionally failed with the error message `Failed to create index...` when creating a unique index. While disgnosing this problem, it was found that the internal error **PREPARE_SEIZE_ERROR** (a temporary error) was reported as an unknown error. Now in such cases, `ndb_restore` retries the creation of the unique index, and **PREPARE_SEIZE_ERROR** is reported as NDB Error 748 `Busy during read of event table`. (Bug #21178339)

  References: See also: Bug #22989944.

- When setting up event logging for `ndb_mgmd` on Windows, MySQL Cluster tries to add a registry key to `HKEY_LOCAL_MACHINE`, which fails if the user does not have access to the registry. In such cases `ndb_mgmd` logged the error `Could neither create or open key`, which is not accurate and which can cause confusion for users who may not realize that file logging is available and being

used. Now in such cases, `ndb_mgmd` logs a warning `Could not create or access the registry key needed for the application to log to the Windows EventLog. Run the application with sufficient privileges once to create the key, or add the key manually, or turn off logging for that application.` An error (as opposed to a warning) is now reported in such cases only if there is no available output at all for `ndb_mgmd` event logging. (Bug #20960839)

- `NdbDictionary` metadata operations had a hard-coded 7-day timeout, which proved to be excessive for short-lived operations such as retrieval of table definitions. This could lead to unnecessary hangs in user applications which were difficult to detect and handle correctly. To help address this issue, timeout behaviour is modified so that read-only or short-duration dictionary interactions have a 2-minute timeout, while schema transactions of potentially long duration retain the existing 7-day timeout.

  Such timeouts are intended as a safety net: In the event of problems, these return control to users, who can then take corrective action. Any reproducible issue with `NdbDictionary` timeouts should be reported as a bug. (Bug #20368354)

- Optimization of signal sending by buffering and sending them periodically, or when the buffer became full, could cause `SUB_GCP_COMPLETE_ACK` signals to be excessively delayed. Such signals are sent for each node and epoch, with a minimum interval of `TimeBetweenEpochs`; if they are not received in time, the `SUMA` buffers can overflow as a result. The overflow caused API nodes to be disconnected, leading to current transactions being aborted due to node failure. This condition made it difficult for long transactions (such as altering a very large table), to be completed. Now in such cases, the `ACK` signal is sent without being delayed. (Bug #18753341)

- An internal function used to validate connections failed to update the connection count when creating a new `Ndb` object. This had the potential to create a new `Ndb` object for every operation validating the connection, which could have an impact on performance, particularly when performing schema operations. (Bug #80750, Bug #22932982)

- When an SQL node was started, and joined the schema distribution protocol, another SQL node, already waiting for a schema change to be distributed, timed out during that wait. This was because the code incorrectly assumed that the new SQL node would also acknowledge the schema distribution even though the new node joined too late to be a participant in it.

  As part of this fix, printouts of schema distribution progress now always print the more significant part of a bitmask before the less significant; formatting of bitmasks in such printouts has also been improved. (Bug #80554, Bug #22842538)

- Settings for the `SchedulerResponsiveness` data node configuration parameter (introduced in MySQL Cluster NDB 7.4.9) were ignored. (Bug #80341, Bug #22712481)

- MySQL Cluster did not compile correctly with Microsoft Visual Studio 2015, due to a change from previous versions in the VS implementation of the `_vsnprintf()` function. (Bug #80276, Bug #22670525)

- When setting CPU spin time, the value was needlessly cast to a boolean internally, so that setting it to any nonzero value yielded an effective value of 1. This issue, as well as the fix for it, apply both to setting the `SchedulerSpinTimer` parameter and to setting `spintime` as part of a `ThreadConfig` parameter value. (Bug #80237, Bug #22647476)

- Processing of local checkpoints was not handled correctly on Mac OS X, due to an uninitialized variable. (Bug #80236, Bug #22647462)

- A logic error in an `if` statement in `storage/ndb/src/kernel/blocks/dbacc/DbaccMain.cpp` rendered useless a check for determining whether `ZREAD_ERROR` should be returned when comparing operations. This was detected when compiling with `gcc` using `-Werror=logical-op`. (Bug #80155, Bug #22601798)

  References: This issue is a regression of: Bug #21285604.

- The `ndb_print_file` utility failed consistently on Solaris 9 for SPARC. (Bug #80096, Bug #22579581)

- Builds with the `-Werror` and `-Wextra` flags (as for release builds) failed on SLES 11. (Bug #79950, Bug #22539531)

- When using `CREATE INDEX` to add an index on either of two `NDB` tables sharing circular foreign keys, the query succeeded but a temporary table was left on disk, breaking the foreign key constraints. This issue was also observed when attempting to create an index on a table in the middle of a chain of foreign keys—that is, a table having both parent and child keys, but on different tables. The problem did not occur when using `ALTER TABLE` to perform the same index creation operation; and subsequent analysis revealed unintended differences in the way such operations were performed by `CREATE INDEX`.

  To fix this problem, we now make sure that operations performed by a `CREATE INDEX` statement are always handled internally in the same way and at the same time that the same operations are handled when performed by `ALTER TABLE` or `DROP INDEX`. (Bug #79156, Bug #22173891)

- `NDB` failed to ignore index prefixes on primary and unique keys, causing `CREATE TABLE` and `ALTER TABLE` statements using them to be rejected. (Bug #78441, Bug #21839248)

- **NDB Cluster APIs:** Executing a transaction with an `NdbIndexOperation` based on an obsolete unique index caused the data node process to fail. Now the index is checked in such cases, and if it cannot be used the transaction fails with an appropriate error. (Bug #79494, Bug #22299443)

Changes in MySQL NDB Cluster 7.4.10 (5.6.28-ndb-7.4.10)

Bugs Fixed

- A serious regression was inadvertently introduced in MySQL Cluster NDB 7.4.8 whereby local checkpoints and thus restarts often took much longer than expected. This occurred due to the fact that the setting for `MaxDiskWriteSpeedOwnRestart` was ignored during restarts and the value of `MaxDiskWriteSpeedOtherNodeRestart`, which is much lower by default than the default for `MaxDiskWriteSpeedOwnRestart`, was used instead. This issue affected restart times and performance only and did not have any impact on normal operations. (Bug #22582233)

Changes in MySQL NDB Cluster 7.4.9 (5.6.28-ndb-7.4.9)

Functionality Added or Changed

- **Important Change:** Previously, the `NDB` scheduler always optimized for speed against throughput in a predetermined manner (this was hard coded); this balance can now be set using the `SchedulerResponsiveness` data node configuration parameter. This parameter accepts an integer in the range of 0-10 inclusive, with 5 as the default. Higher values provide better response times relative to throughput. Lower values provide increased throughput, but impose longer response times. (Bug #78531, Bug #21889312)

- Added the `tc_time_track_stats` table to the `ndbinfo` information database. This table provides time-tracking information relating to transactions, key operations, and scan operations performed by `NDB`. (Bug #78533, Bug #21889652)

Bugs Fixed

- **Important Change:** A fix made in MySQL Cluster NDB 7.3.11 and MySQL Cluster NDB 7.4.8 caused `ndb_restore` to perform unique key checks even when operating in modes which do not restore data, such as when using the program's `--restore_epoch` or `--print_data` option.

  That change in behavior caused existing valid backup routines to fail; to keep this issue from affecting this and future releases, the previous fix has been reverted. This means that the requirement added in those versions that ndb_restore be run `--disable-indexes` or `--rebuild-indexes` when used on tables containing unique indexes is also lifted. (Bug #22345748)

References: See also: Bug #22329365. Reverted patches: Bug #57782, Bug #11764893.

- **Important Change:** Users can now set the number and length of connection timeouts allowed by most `NDB` programs with the `--connect-retries` and `--connect-retry-delay` command line options introduced for the programs in this release. For `ndb_mgm`, `--connect-retries` supersedes the existing `--try-reconnect` option. (Bug #57576, Bug #11764714)

- In debug builds, a `WAIT_EVENT` while polling caused excessive logging to stdout. (Bug #22203672)

- When executing a schema operation such as `CREATE TABLE` on a MySQL Cluster with multiple SQL nodes, it was possible for the SQL node on which the operation was performed to time out while waiting for an acknowledgement from the others. This could occur when different SQL nodes had different settings for `--ndb-log-updated-only`, `--ndb-log-update-as-write`, or other `mysqld` options effecting binary logging by `NDB`.

  This happened due to the fact that, in order to distribute schema changes between them, all SQL nodes subscribe to changes in the `ndb_schema` system table, and that all SQL nodes are made aware of each others subscriptions by subscribing to `TE_SUBSCRIBE` and `TE_UNSUBSCRIBE` events. The names of events to subscribe to are constructed from the table names, adding `REPL$` or `REPLF$` as a prefix. `REPLF$` is used when full binary logging is specified for the table. The issue described previously arose because different values for the options mentioned could lead to different events being subscribed to by different SQL nodes, meaning that all SQL nodes were not necessarily aware of each other, so that the code that handled waiting for schema distribution to complete did not work as designed.

  To fix this issue, MySQL Cluster now treats the `ndb_schema` table as a special case and enforces full binary logging at all times for this table, independent of any settings for `mysqld` binary logging options. (Bug #22174287, Bug #79188)

- Attempting to create an `NDB` table having greater than the maximum supported combined width for all `BIT` columns (4096) caused data node failure when these columns were defined with `COLUMN_FORMAT DYNAMIC`. (Bug #21889267)

- Creating a table with the maxmimum supported number of columns (512) all using `COLUMN_FORMAT DYNAMIC` led to data node failures. (Bug #21863798)

- In certain cases, a cluster failure (error 4009) was reported as `Unknown error code`. (Bug #21837074)

- For a timeout in `GET_TABINFOREQ` while executing a `CREATE INDEX` statement, mysqld returned Error 4243 (`Index not found`) instead of the expected Error 4008 (`Receive from NDB failed`).

  The fix for this bug also fixes similar timeout issues for a number of other signals that are sent the `DBDICT` kernel block as part of DDL operations, including `ALTER_TAB_REQ`, `CREATE_INDX_REQ`, `DROP_FK_REQ`, `DROP_INDX_REQ`, `INDEX_STAT_REQ`, `DROP_FILE_REQ`, `CREATE_FILEGROUP_REQ`, `DROP_FILEGROUP_REQ`, `CREATE_EVENT`, `WAIT_GCP_REQ`, `DROP_TAB_REQ`, and `LIST_TABLES_REQ`, as well as several internal functions used in handling `NDB` schema operations. (Bug #21277472)

  References: See also: Bug #20617891, Bug #20368354, Bug #19821115.

- Using `ndb_mgm STOP -f` to force a node shutdown even when it triggered a complete shutdown of the cluster, it was possible to lose data when a sufficient number of nodes were shut down, triggering a cluster shutodwn, and the timing was such that `SUMA` handovers had been made to nodes already in the process of shutting down. (Bug #17772138)

- The internal `NdbEventBuffer::set_total_buckets()` method calculated the number of remaining buckets incorrectly. This caused any incomplete epoch to be prematurely completed when the `SUB_START_CONF` signal arrived out of order. Any events belonging to this epoch arriving later

were then ignored, and so effectively lost, which resulted in schema changes not being distributed correctly among SQL nodes. (Bug #79635, Bug #22363510)

- Compilation of MySQL Cluster failed on SUSE Linux Enterprise Server 12. (Bug #79429, Bug #22292329)

- Schema events were appended to the binary log out of order relative to non-schema events. This was caused by the fact that the binary log injector did not properly handle the case where schema events and non-schema events were from different epochs.

  This fix modifies the handling of events from the two schema and non-schema event streams such that events are now always handled one epoch at a time, starting with events from the oldest available epoch, without regard to the event stream in which they occur. (Bug #79077, Bug #22135584, Bug #20456664)

- When executed on an `NDB` table, `ALTER TABLE ... DROP INDEX` made changes to an internal array referencing the indexes before the index was actually dropped, and did not revert these changes in the event that the drop was not completed. One effect of this was that, after attempting to drop an index on which there was a foreign key dependency, the expected error referred to the wrong index, and subsequent attempts using SQL to modify indexes of this table failed. (Bug #78980, Bug #22104597)

- `NDB` failed during a node restart due to the status of the current local checkpoint being set but not as active, even though it could have other states under such conditions. (Bug #78780, Bug #21973758)

- `ndbmtd` checked for signals being sent only after a full cycle in `run_job_buffers`, which is performed for all job buffer inputs. Now this is done as part of `run_job_buffers` itself, which avoids executing for extended periods of time without sending to other nodes or flushing signals to other threads. (Bug #78530, Bug #21889088)

- The value set for `spintime` by the `ThreadConfig` parameter was not calculated correctly, causing the spin to continue for longer than actually specified. (Bug #78525, Bug #21886476)

- When `NDBFS` completed file operations, the method it employed for waking up the main thread worked effectively on Linux/x86 platforms, but not on some others, including OS X, which could lead to unnecessary slowdowns on those platforms. (Bug #78524, Bug #21886157)

- **NDB Disk Data:** A unique index on a column of an `NDB` table is implemented with an associated internal ordered index, used for scanning. While dropping an index, this ordered index was dropped first, followed by the drop of the unique index itself. This meant that, when the drop was rejected due to (for example) a constraint violation, the statement was rejected but the associated ordered index remained deleted, so that any subsequent operation using a scan on this table failed. We fix this problem by causing the unique index to be removed first, before removing the ordered index; removal of the related ordered index is no longer performed when removal of a unique index fails. (Bug #78306, Bug #21777589)

- **NDB Cluster APIs:** The binary log injector did not work correctly with `TE_INCONSISTENT` event type handling by `Ndb::nextEvent()`. (Bug #22135541)

  References: See also: Bug #20646496.

- **NDB Cluster APIs:** `Ndb::pollEvents()` and `pollEvents2()` were slow to receive events, being dependent on other client threads or blocks to perform polling of transporters on their behalf. This fix allows a client thread to perform its own transporter polling when it has to wait in either of these methods.

  Introduction of transporter polling also revealed a problem with missing mutex protection in the `ndbcluster_binlog` handler, which has been added as part of this fix. (Bug #79311, Bug #20957068, Bug #22224571)

- **NDB Cluster APIs:** Garbage collection is performed on several objects in the implementation of `NdbEventOperation`, based on which GCIs have been consumed by clients, including those that

have been dropped by `Ndb::dropEventOperation()`. In this implementation, the assumption was made that the global checkpoint index (GCI) is always monotonically increasing, although this is not the case during an initial restart, when the GCI is reset. This could lead to event objects in the NDB API being released prematurely or not at all, in the latter case causing a resource leak.

To prevent this from happening, the NDB event object's implementation now tracks, internally, both the GCI and the generation of the GCI; the generation is incremented whenever the node process is restarted, and this value is now used to provide a monotonically increasing sequence. (Bug #73781, Bug #21809959)

Changes in MySQL NDB Cluster 7.4.8 (5.6.27-ndb-7.4.8)

Functionality Added or Changed

- **Incompatible Change:** The changes listed here follow up and build further on work done in MySQL Cluster NDB 7.4.7 to improve handling of local checkpoints (LCPs) under conditions of insert overload:

  - Changes have been made in the minimum values for a number of parameters applying to data buffers for backups and LCPs. These parameters, listed here, can no longer be set so as to make the system impossible to run:

    - `BackupDataBufferSize`: minimum increased from 0 to 2M.

    - `BackupLogBufferSize`: minimum increased from 0 to 2M.

    - `BackupWriteSize`: minimum increased from 2K to 32K.

    - `BackupMaxWriteSize`: minimum increased from 2K to 256K.

    In addition, the `BackupMemory` data node parameter is now deprecated and subject to removal in a future version of MySQL Cluster. Use `BackupDataBufferSize` and `BackupLogBufferSize` instead.

  - When a backup was unsuccessful due to insufficient resources, a subsequent retry worked only for those parts of the backup that worked in the same thread, since delayed signals are only supported in the same thread. Delayed signals are no longer sent to other threads in such cases.

  - An instance of an internal list object used in searching for queued scans was not actually destroyed before calls to functions that could manipulate the base object used to create it.

  - ACC scans were queued in the category of range scans, which could lead to starting an ACC scan when `DBACC` had no free slots for scans. We fix this by implementing a separate queue for ACC scans.

  (Bug #76890, Bug #20981491, Bug #77597, Bug #21362758, Bug #77612, Bug #21370839)

  References: See also: Bug #76742, Bug #20904721.

- When the `--database` option has not been specified for `ndb_show_tables`, and no tables are found in the `TEST_DB` database, an appropriate warning message is now issued. (Bug #50633, Bug #11758430)

Bugs Fixed

- **Important Change:** When `ndb_restore` was run without `--disable-indexes` or `--rebuild-indexes` on a table having a unique index, it was possible for rows to be restored in an order that resulted in duplicate values, causing it to fail with duplicate key errors. Running `ndb_restore` on such a table now requires using at least one of these options; failing to do so now results in an error. (Bug #57782, Bug #11764893)

  References: See also: Bug #22329365, Bug #22345748.

- **Important Change; NDB Cluster APIs:** The MGM API error-handling functions `ndb_mgm_get_latest_error()`, `ndb_mgm_get_latest_error_msg()`, and `ndb_mgm_get_latest_error_desc()` each failed when used with a `NULL` handle. You should note that, although these functions are now null-safe, values returned in this case are arbitrary and not meaningful. (Bug #78130, Bug #21651706)

- `mysql_upgrade` failed when performing an upgrade from MySQL Cluster NDB 7.2 to MySQL Cluster NDB 7.4. The root cause of this issue was an accidental duplication of code in `mysql_fix_privilege_tables.sql` that caused `ndbinfo_offline` mode to be turned off too early, which in turn led a subsequent `CREATE VIEW` statement to fail. (Bug #21841821)

- `ClusterMgr` is a internal component of NDB API and `ndb_mgmd` processes, part of `TransporterFacade`—which in turn is a wrapper around the transporter registry—and shared with data nodes. This component is responsible for a number of tasks including connection setup requests; sending and monitoring of heartbeats; provision of node state information; handling of cluster disconnects and reconnects; and forwarding of cluster state indicators. `ClusterMgr` maintains a count of live nodes which is incremented on receiving a report of a node having connected (`reportConnected()` method call), and decremented on receiving a report that a node has disconnected (`reportDisconnected()`) from `TransporterRegistry`. This count is checked within `reportDisconnected()` to verify that is it greater than zero.

  The issue addressed here arose when node connections were very brief due to send buffer exhaustion (among other potential causes) and the check just described failed. This occurred because, when a node did not fully connect, it was still possible for the connection attempt to trigger a `reportDisconnected()` call in spite of the fact that the connection had not yet been reported to `ClusterMgr`; thus, the pairing of `reportConnected()` and `reportDisconnected()` calls was not guaranteed, which could cause the count of connected nodes to be set to zero even though there remained nodes that were still in fact connected, causing node crashes with debug builds of MySQL Cluster, and potential errors or other adverse effects with release builds.

  To fix this issue, `ClusterMgr::reportDisconnected()` now verifies that a disconnected node had actually finished connecting completely before checking and decrementing the number of connected nodes. (Bug #21683144, Bug #22016081)

  References: See also: Bug #21664515, Bug #21651400.

- To reduce the possibility that a node's loopback transporter becomes disconnected from the transporter registry by `reportError()` due to send buffer exhaustion (implemented by the fix for Bug #21651400), a portion of the send buffer is now reserved for the use of this transporter. (Bug #21664515, Bug #22016081)

  References: See also: Bug #21651400, Bug #21683144.

- The loopback transporter is similar to the TCP transporter, but is used by a node to send signals to itself as part of many internal operations. Like the TCP transporter, it could be disconnected due to certain conditions including send buffer exhaustion, but this could result in blocking of `TransporterFacade` and so cause multiple issues within an `ndb_mgmd` or API node process. To prevent this, a node whose loopback transporter becomes disconnected is now simply shut down, rather than allowing the node process to hang. (Bug #21651400, Bug #22016081)

  References: See also: Bug #21683144, Bug #21664515.

- The internal `NdbEventBuffer` object's active subscriptions count (`m_active_op_count`) could be decremented more than once when stopping a subscription when this action failed, for example, due to a busy server and was retried. Decrementing of this count could also fail when communication with the data node failed, such as when a timeout occurred. (Bug #21616263)

  References: This issue is a regression of: Bug #20575424, Bug #20561446.

- In some cases, the management server daemon failed on startup without reporting the reason. Now when `ndb_mgmd` fails to start due to an error, the error message is printed to `stderr`. (Bug #21571055)

- In a MySQL Cluster with multiple LDM instances, all instances wrote to the node log, even inactive instances on other nodes. During restarts, this caused the log to be filled with messages from other nodes, such as the messages shown here:

```
2015-06-24 00:20:16 [ndbd] INFO     -- We are adjusting Max Disk Write Speed,
a restart is ongoing now
...
2015-06-24 01:08:02 [ndbd] INFO     -- We are adjusting Max Disk Write Speed,
no restarts ongoing anymore
```

  Now this logging is performed only by the active LDM instance. (Bug #21362380)

- Backup block states were reported incorrectly during backups. (Bug #21360188)

  References: See also: Bug #20204854, Bug #21372136.

- Added the `BackupDiskWriteSpeedPct` data node parameter. Setting this parameter causes the data node to reserve a percentage of its maximum write speed (as determined by the value of `MaxDiskWriteSpeed`) for use in local checkpoints while performing a backup. `BackupDiskWriteSpeedPct` is interpreted as a percentage which can be set between 0 and 90 inclusive, with a default value of 50. (Bug #20204854)

  References: See also: Bug #21372136.

- When a data node is known to have been alive by other nodes in the cluster at a given global checkpoint, but its `sysfile` reports a lower GCI, the higher GCI is used to determine which global checkpoint the data node can recreate. This caused problems when the data node being started had a clean file system (GCI = 0), or when it was more than more global checkpoint behind the other nodes.

  Now in such cases a higher GCI known by other nodes is used only when it is at most one GCI ahead. (Bug #19633824)

  References: See also: Bug #20334650, Bug #21899993. This issue is a regression of: Bug #29167.

- When restoring a specific database or databases with the `--include-databases` or `--exclude-databases` option, `ndb_restore` attempted to apply foreign keys on tables in databases which were not among those being restored. (Bug #18560951)

- After restoring the database schema from backup using `ndb_restore`, auto-discovery of restored tables in transactions having multiple statements did not work correctly, resulting in `Deadlock found when trying to get lock; try restarting transaction` errors.

  This issue was encountered both in the `mysql` client, as well as when such transactions were executed by application programs using Connector/J and possibly other MySQL APIs.

  Prior to upgrading, this issue can be worked around by executing `SELECT TABLE_NAME, TABLE_SCHEMA FROM INFORMATION_SCHEMA.TABLES WHERE ENGINE = 'NDBCLUSTER'` on all SQL nodes following the restore operation, before executing any other statements. (Bug #18075170)

- `ndb_desc` used with the `--extra-partition-info` and `--blob-info` options failed when run against a table containing one or more `TINYBLOB`. columns. (Bug #14695968)

- Operations relating to global checkpoints in the internal event data buffer could sometimes leak memory. (Bug #78205, Bug #21689380)

  References: See also: Bug #76165, Bug #20651661.

- Trying to create an `NDB` table with a composite foreign key referencing a composite primary key of the parent table failed when one of the columns in the composite foreign key was the table's primary key and in addition this column also had a unique key. (Bug #78150, Bug #21664899)

- When attempting to enable index statistics, creation of the required system tables, events and event subscriptions often fails when multiple `mysqld` processes using index statistics are started concurrently in conjunction with starting, restarting, or stopping the cluster, or with node failure handling. This is normally recoverable, since the affected `mysqld` process or processes can (and do) retry these operations shortly thereafter. For this reason, such failures are no longer logged as warnings, but merely as informational events. (Bug #77760, Bug #21462846)

- Adding a unique key to an `NDB` table failed when the table already had a foreign key. Prior to upgrading, you can work around this issue by creating the unique key first, then adding the foreign key afterwards, using a separate `ALTER TABLE` statement. (Bug #77457, Bug #20309828)

- **NDB Cluster APIs:** While executing `dropEvent()`, if the coordinator `DBDICT` failed after the subscription manager (`SUMA` block) had removed all subscriptions but before the coordinator had deleted the event from the system table, the dropped event remained in the table, causing any subsequent drop or create event with the same name to fail with `NDB` error 1419 `Subscription already dropped` or error 746 `Event name already exists`. This occurred even when calling `dropEvent()` with a nonzero force argument.

  Now in such cases, error 1419 is ignored, and `DBDICT` deletes the event from the table. (Bug #21554676)

- **NDB Cluster APIs:** If the total amount of memory allocated for the event buffer exceeded approximately 40 MB, the calculation of memory usage percentages could overflow during computation. This was due to the fact that the associated routine used 32-bit arithmetic; this has now been changed to use `Uint64` values instead. (Bug #78454, Bug #21847552)

- **NDB Cluster APIs:** The `nextEvent2()` method continued to return exceptional events such as `TE_EMPTY`, `TE_INCONSISTENT`, and `TE_OUT_OF_MEMORY` for event operations which already had been dropped. (Bug #78167, Bug #21673318)

- **NDB Cluster APIs:** After the initial restart of a node following a cluster failure, the cluster failure event added as part of the restart process was deleted when an event that existed prior to the restart was later deleted. This meant that, in such cases, an Event API client had no way of knowing that failure handling was needed. In addition, the GCI used for the final cleanup of deleted event operations, performed by `pollEvents()` and `nextEvent()` when these methods have consumed all available events, was lost. (Bug #78143, Bug #21660947)

- **NDB Cluster APIs:** The internal value representing the latest global checkpoint was not always updated when a completed epoch of event buffers was inserted into the event queue. This caused subsequent calls to `Ndb::pollEvents()` and `pollEvents2()` to fail when trying to obtain the correct GCI for the events available in the event buffers. This could also result in later calls to `nextEvent()` or `nextEvent2()` seeing events that had not yet been discovered. (Bug #78129, Bug #21651536)

Changes in MySQL NDB Cluster 7.4.7 (5.6.25-ndb-7.4.7)

Functionality Added or Changed

- Deprecated MySQL Cluster node configuration parameters are now indicated as such by `ndb_config --configinfo --xml`. For each parameter currently deprecated, the corresponding `<param/>` tag in the XML output now includes the attribute `deprecated="true"`. (Bug #21127135)

- A number of improvements, listed here, have been made with regard to handling issues that could arise when an overload arose due to a great number of inserts being performed during a local checkpoint (LCP):

- Failures sometimes occurred during restart processing when trying to execute the undo log, due to a problem with finding the end of the log. This happened when there remained unwritten pages at the end of the first undo file when writing to the second undo file, which caused the execution of undo logs in reverse order and so execute old or even nonexistent log records.

  This is fixed by ensuring that execution of the undo log begins with the proper end of the log, and, if started earlier, that any unwritten or faulty pages are ignored.

- It was possible to fail during an LCP, or when performing a `COPY_FRAGREQ`, due to running out of operation records. We fix this by making sure that LCPs and `COPY_FRAG` use resources reserved for operation records, as was already the case with scan records. In addition, old code for ACC operations that was no longer required but that could lead to failures was removed.

- When an LCP was performed while loading a table, it was possible to hit a livelock during LCP scans, due to the fact that that each record that was inserted into new pages after the LCP had started had its `LCP_SKIP` flag set. Such records were discarded as intended by the LCP scan, but when inserts occurred faster than the LCP scan could discard records, the scan appeared to hang. As part of this issue, the scan failed to report any progress to the LCP watchdog, which after 70 seconds of livelock killed the process. This issue was observed when performing on the order of 250000 inserts per second over an extended period of time (120 seconds or more), using a single LDM.

  This part of the fix makes a number of changes, listed here:

  - We now ensure that pages created after the LCP has started are not included in LCP scans; we also ensure that no records inserted into those pages have their `LCP_SKIP` flag set.

  - Handling of the scan protocol is changed such that a certain amount of progress is made by the LCP regardless of load; we now report progress to the LCP watchdog so that we avoid failure in in the event that an LCP is making progress but not writing any records.

  - We now take steps to guarantee that LCP scans proceed more quickly than inserts can occur, by ensuring that scans are prioritized this scanning activity, and thus, that the LCP is in fact (eventually) completed.

  - In addition, scanning is made more efficient, by prefetching tuples; this helps avoid stalls while fetching memory in the CPU.

- Row checksums for preventing data corruption now include the tuple header bits.

  (Bug #76373, Bug #20727343, Bug #76741, Bug #69994, Bug #20903880, Bug #76742, Bug #20904721, Bug #76883, Bug #20980229)

Bugs Fixed

- **Incompatible Change; NDB Cluster APIs:** The `pollEvents2()` method now returns -1, indicating an error, whenever a negative value is used for the time argument. (Bug #20762291)

- **Important Change; NDB Cluster APIs:** The `Ndb::getHighestQueuedEpoch()` method returned the greatest epoch in the event queue instead of the greatest epoch found after calling `pollEvents2()`. (Bug #20700220)

- **Important Change; NDB Cluster APIs:** `Ndb::pollEvents()` is now compatible with the `TE_EMPTY`, `TE_INCONSISTENT`, and `TE_OUT_OF_MEMORY` event types introduced in MySQL Cluster NDB 7.4.3. For detailed information about this change, see the description of this method in the *MySQL Cluster API Developer Guide*. (Bug #20646496)

- **Important Change; NDB Cluster APIs:** Added the method `Ndb::isExpectingHigherQueuedEpochs()` to the NDB API to detect when additional, newer event epochs were detected by `pollEvents2()`.

The behavior of `Ndb::pollEvents()` has also been modified such that it now returns **`NDB_FAILURE_GCI`** (equal to `~(Uint64) 0`) when a cluster failure has been detected. (Bug #18753887)

- After restoring the database metadata (but not any data) by running `ndb_restore --restore_meta` (or `-m`), SQL nodes would hang while trying to `SELECT` from a table in the database to which the metadata was restored. In such cases the attempt to query the table now fails as expected, since the table does not actually exist until `ndb_restore` is executed with `--restore_data` (`-r`). (Bug #21184102)

  References: See also: Bug #16890703.

- When a great many threads opened and closed blocks in the NDB API in rapid succession, the internal `close_clnt()` function synchronizing the closing of the blocks waited an insufficiently long time for a self-signal indicating potential additional signals needing to be processed. This led to excessive CPU usage by `ndb_mgmd`, and prevented other threads from opening or closing other blocks. This issue is fixed by changing the function polling call to wait on a specific condition to be woken up (that is, when a signal has in fact been executed). (Bug #21141495)

- Previously, multiple send threads could be invoked for handling sends to the same node; these threads then competed for the same send lock. While the send lock blocked the additional send threads, work threads could be passed to other nodes.

  This issue is fixed by ensuring that new send threads are not activated while there is already an active send thread assigned to the same node. In addition, a node already having an active send thread assigned to it is no longer visible to other, already active, send threads; that is, such a node is longer added to the node list when a send thread is currently assigned to it. (Bug #20954804, Bug #76821)

- Queueing of pending operations when the redo log was overloaded (`DefaultOperationRedoProblemAction` API node configuration parameter) could lead to timeouts when data nodes ran out of redo log space (`P_TAIL_PROBLEM` errors). Now when the redo log is full, the node aborts requests instead of queuing them. (Bug #20782580)

  References: See also: Bug #20481140.

- An `NDB` event buffer can be used with an `Ndb` object to subscribe to table-level row change event streams. Users subscribe to an existing event; this causes the data nodes to start sending event data signals (`SUB_TABLE_DATA`) and epoch completion signals (`SUB_GCP_COMPLETE`) to the `Ndb` object. `SUB_GCP_COMPLETE_REP` signals can arrive for execution in concurrent receiver thread before completion of the internal method call used to start a subscription.

  Execution of `SUB_GCP_COMPLETE_REP` signals depends on the total number of `SUMA` buckets (sub data streams), but this may not yet have been set, leading to the present issue, when the counter used for tracking the `SUB_GCP_COMPLETE_REP` signals (`TOTAL_BUCKETS_INIT`) was found to be set to erroneous values. Now `TOTAL_BUCKETS_INIT` is tested to be sure it has been set correctly before it is used. (Bug #20575424, Bug #76255)

  References: See also: Bug #20561446, Bug #21616263.

- `NDB` statistics queries could be delayed by the error delay set for `ndb_index_stat_option` (default 60 seconds) when the index that was queried had been marked with internal error. The same underlying issue could also cause `ANALYZE TABLE` to hang when executed against an `NDB` table having multiple indexes where an internal error occured on one or more but not all indexes.

  Now in such cases, any existing statistics are returned immediately, without waiting for any additonal statistics to be discovered. (Bug #20553313, Bug #20707694, Bug #76325)

- The multi-threaded scheduler sends to remote nodes either directly from each worker thread or from dedicated send threadsL, depending on the cluster's configuration. This send might transmit

all, part, or none of the available data from the send buffers. While there remained pending send data, the worker or send threads continued trying to send in a loop. The actual size of the data sent in the most recent attempt to perform a send is now tracked, and used to detect lack of send progress by the send or worker threads. When no progress has been made, and there is no other work outstanding, the scheduler takes a 1 millisecond pause to free up the CPU for use by other threads. (Bug #18390321)

References: See also: Bug #20929176, Bug #20954804.

- In some cases, attempting to restore a table that was previously backed up failed with a `File Not Found` error due to a missing table fragment file. This occurred as a result of the NDB kernel `BACKUP` block receiving a `Busy` error while trying to obtain the table description, due to other traffic from external clients, and not retrying the operation.

  The fix for this issue creates two separate queues for such requests—one for internal clients such as the `BACKUP` block or `ndb_restore`, and one for external clients such as API nodes—and prioritizing the internal queue.

  Note that it has always been the case that external client applications using the NDB API (including MySQL applications running against an SQL node) are expected to handle `Busy` errors by retrying transactions at a later time; this expectation is *not* changed by the fix for this issue. (Bug #17878183)

  References: See also: Bug #17916243.

- On startup, API nodes (including `mysqld` processes running as SQL nodes) waited to connect with data nodes that had not yet joined the cluster. Now they wait only for data nodes that have actually already joined the cluster.

  In the case of a new data node joining an existing cluster, API nodes still try to connect with the new data node within `HeartbeatIntervalDbApi` milliseconds. (Bug #17312761)

- In some cases, the `DBDICT` block failed to handle repeated `GET_TABINFOREQ` signals after the first one, leading to possible node failures and restarts. This could be observed after setting a sufficiently high value for `MaxNoOfExecutionThreads` and low value for `LcpScanProgressTimeout`. (Bug #77433, Bug #21297221)

- Client lookup for delivery of API signals to the correct client by the internal `TransporterFacade::deliver_signal()` function had no mutex protection, which could cause issues such as timeouts encountered during testing, when other clients connected to the same `TransporterFacade`. (Bug #77225, Bug #21185585)

- It was possible to end up with a lock on the send buffer mutex when send buffers became a limiting resource, due either to insufficient send buffer resource configuration, problems with slow or failing communications such that all send buffers became exhausted, or slow receivers failing to consume what was sent. In this situation worker threads failed to allocate send buffer memory for signals, and attempted to force a send in order to free up space, while at the same time the send thread was busy trying to send to the same node or nodes. All of these threads competed for taking the send buffer mutex, which resulted in the lock already described, reported by the watchdog as `Stuck in Send`. This fix is made in two parts, listed here:

  1. The send thread no longer holds the global send thread mutex while getting the send buffer mutex; it now releases the global mutex prior to locking the send buffer mutex. This keeps worker threads from getting stuck in send in such cases.

  2. Locking of the send buffer mutex done by the send threads now uses a try-lock. If the try-lock fails, the node to make the send to is reinserted at the end of the list of send nodes in order to be retried later. This removes the `Stuck in Send` condition for the send threads.

  (Bug #77081, Bug #21109605)

- **NDB Cluster APIs:** Added the `Column::getSizeInBytesForRecord()` method, which returns the size required for a column by an `NdbRecord`, depending on the column's type (text/blob, or other). (Bug #21067283)

- **NDB Cluster APIs:** `NdbEventOperation::isErrorEpoch()` incorrectly returned `false` for the `TE_INCONSISTENT` table event type (see Event::TableEvent). This caused a subsequent call to `getEventType()` to fail. (Bug #20729091)

- **NDB Cluster APIs:** Creation and destruction of `Ndb_cluster_connection` objects by multiple threads could make use of the same application lock, which in some cases led to failures in the global dictionary cache. To alleviate this problem, the creation and destruction of several internal NDB API objects have been serialized. (Bug #20636124)

- **NDB Cluster APIs:** A number of timeouts were not handled correctly in the NDB API.

  (Bug #20617891)

- **NDB Cluster APIs:** When an `Ndb` object created prior to a failure of the cluster was reused, the event queue of this object could still contain data node events originating from before the failure. These events could reference "old" epochs (from before the failure occurred), which in turn could violate the assumption made by the `nextEvent()` method that epoch numbers always increase. This issue is addressed by explicitly clearing the event queue in such cases. (Bug #18411034)

  References: See also: Bug #20888668.

Changes in MySQL NDB Cluster 7.4.6 (5.6.24-ndb-7.4.6)

Bugs Fixed

- During backup, loading data from one SQL node followed by repeated `DELETE` statements on the tables just loaded from a different SQL node could lead to data node failures. (Bug #18949230)

- When an instance of `NdbEventBuffer` was destroyed, any references to GCI operations that remained in the event buffer data list were not freed. Now these are freed, and items from the event bufer data list are returned to the free list when purging GCI containers. (Bug #76165, Bug #20651661)

- When a bulk delete operation was committed early to avoid an additional round trip, while also returning the number of affected rows, but failed with a timeout error, an SQL node performed no verification that the transaction was in the Committed state. (Bug #74494, Bug #20092754)

  References: See also: Bug #19873609.

Changes in MySQL NDB Cluster 7.4.5 (5.6.23-ndb-7.4.5)

Bugs Fixed

- **Important Change:** The maximum failure time calculation used to ensure that normal node failure handling mechanisms are given time to handle survivable cluster failures (before global checkpoint watchdog mechanisms start to kill nodes due to GCP delays) was excessively conservative, and neglected to consider that there can be at most *number_of_data_nodes* / `NoOfReplicas` node failures before the cluster can no longer survive. Now the value of `NoOfReplicas` is properly taken into account when performing this calculation.

  This fix adds the `TimeBetweenGlobalCheckpointsTimeout` data node configuration parameter, which makes the minimum timeout between global checkpoints settable by the user. This timeout was previously fixed internally at 120000 milliseconds, which is now the default value for this parameter. (Bug #20069617, Bug #20069624)

  References: See also: Bug #19858151, Bug #20128256, Bug #20135976.

- In the event of a node failure during an initial node restart followed by another node start, the restart of the affected node could hang with a `START_INFOREQ` that occurred while invalidation of local checkpoints was still ongoing. (Bug #20546157, Bug #75916)

  References: See also: Bug #34702.

- It was found during testing that problems could arise when the node registered as the arbitrator disconnected or failed during the arbitration process.

  In this situation, the node requesting arbitration could never receive a positive acknowledgement from the registered arbitrator; this node also lacked a stable set of members and could not initiate selection of a new arbitrator.

  Now in such cases, when the arbitrator fails or loses contact during arbitration, the requesting node immediately fails rather than waiting to time out. (Bug #20538179)

- `DROP DATABASE` failed to remove the database when the database directory contained a `.ndb` file which had no corresponding table in `NDB`. Now, when executing `DROP DATABASE`, `NDB` performs an check specifically for leftover `.ndb` files, and deletes any that it finds. (Bug #20480035)

  References: See also: Bug #44529.

- When performing a restart, it was sometimes possible to find a log end marker which had been written by a previous restart, and that should have been invalidated. Now when searching for the last page to invalidate, the same search algorithm is used as when searching for the last page of the log to read. (Bug #76207, Bug #20665205)

- During a node restart, if there was no global checkpoint completed between the `START_LCP_REQ` for a local checkpoint and its `LCP_COMPLETE_REP` it was possible for a comparison of the LCP ID sent in the `LCP_COMPLETE_REP` signal with the internal value `SYSFILE->latestLCP_ID` to fail. (Bug #76113, Bug #20631645)

- When sending `LCP_FRAG_ORD` signals as part of master takeover, it is possible that the master not is not synchronized with complete accuracy in real time, so that some signals must be dropped. During this time, the master can send a `LCP_FRAG_ORD` signal with its `lastFragmentFlag` set even after the local checkpoint has been completed. This enhancement causes this flag to persist until the statrt of the next local checkpoint, which causes these signals to be dropped as well.

  This change affects `ndbd` only; the issue described did not occur with `ndbmtd`. (Bug #75964, Bug #20567730)

- When reading and copying transporter short signal data, it was possible for the data to be copied back to the same signal with overlapping memory. (Bug #75930, Bug #20553247)

- NDB node takeover code made the assumption that there would be only one takeover record when starting a takeover, based on the further assumption that the master node could never perform copying of fragments. However, this is not the case in a system restart, where a master node can have stale data and so need to perform such copying to bring itself up to date. (Bug #75919, Bug #20546899)

- **NDB Cluster APIs:** A scan operation, whether it is a single table scan or a query scan used by a pushed join, stores the result set in a buffer. This maximum size of this buffer is calculated and preallocated before the scan operation is started. This buffer may consume a considerable amount of memory; in some cases we observed a 2 GB buffer footprint in tests that executed 100 parallel scans with 2 single-threaded (`ndbd`) data nodes. This memory consumption was found to scale linearly with additional fragments.

  A number of root causes, listed here, were discovered that led to this problem:

- Result rows were unpacked to full `NdbRecord` format before they were stored in the buffer. If only some but not all columns of a table were selected, the buffer contained empty space (essentially wasted).

- Due to the buffer format being unpacked, `VARCHAR` and `VARBINARY` columns always had to be allocated for the maximum size defined for such columns.

- `BatchByteSize` and `MaxScanBatchSize` values were not taken into consideration as a limiting factor when calculating the maximum buffer size.

These issues became more evident in NDB 7.2 and later MySQL Cluster release series. This was due to the fact buffer size is scaled by `BatchSize`, and that the default value for this parameter was increased fourfold (from 64 to 256) beginning with MySQL Cluster NDB 7.2.1.

This fix causes result rows to be buffered using the packed format instead of the unpacked format; a buffered scan result row is now not unpacked until it becomes the current row. In addition, `BatchByteSize` and `MaxScanBatchSize` are now used as limiting factors when calculating the required buffer size.

Also as part of this fix, refactoring has been done to separate handling of buffered (packed) from handling of unbuffered result sets, and to remove code that had been unused since NDB 7.0 or earlier. The `NdbRecord` class declaration has also been cleaned up by removing a number of unused or redundant member variables. (Bug #73781, Bug #75599, Bug #19631350, Bug #20408733)

Changes in MySQL NDB Cluster 7.4.4 (5.6.23-ndb-7.4.4)

Bugs Fixed

- When upgrading a MySQL Cluster from NDB 7.3 to NDB 7.4, the first data node started with the NDB 7.4 data node binary caused the master node (still running NDB 7.3) to fail with Error 2301, then itself failed during Start Phase 5. (Bug #20608889)

- A memory leak in NDB event buffer allocation caused an event to be leaked for each epoch. (Due to the fact that an SQL node uses 3 event buffers, each SQL node leaked 3 events per epoch.) This meant that a MySQL Cluster `mysqld` leaked an amount of memory that was inversely proportional to the size of `TimeBetweenEpochs`—that is, the smaller the value for this parameter, the greater the amount of memory leaked per unit of time. (Bug #20539452)

- The values of the `Ndb_last_commit_epoch_server` and `Ndb_last_commit_epoch_session` status variables were incorrectly reported on some platforms. To correct this problem, these values are now stored internally as `long long`, rather than `long`. (Bug #20372169)

- When restoring a MySQL Cluster from backup, nodes that failed and were restarted during restoration of another node became unresponsive, which subsequently caused `ndb_restore` to fail and exit. (Bug #20069066)

- When a data node fails or is being restarted, the remaining nodes in the same nodegroup resend to subscribers any data which they determine has not already been sent by the failed node. Normally, when a data node (actually, the `SUMA` kernel block) has sent all data belonging to an epoch for which it is responsible, it sends a `SUB_GCP_COMPLETE_REP` signal, together with a count, to all subscribers, each of which responds with a `SUB_GCP_COMPLETE_ACK`. When `SUMA` receives this acknowledgment from all subscribers, it reports this to the other nodes in the same nodegroup so that they know that there is no need to resend this data in case of a subsequent node failure. If a node failed before all subscribers sent this acknowledgement but before all the other nodes in the same nodegroup received it from the failing node, data for some epochs could be sent (and reported as complete) twice, which could lead to an unplanned shutdown.

The fix for this issue adds to the count reported by `SUB_GCP_COMPLETE_ACK` a list of identifiers which the receiver can use to keep track of which buckets are completed and to ignore any duplicate reported for an already completed bucket. (Bug #17579998)

- The `ndbinfo.restart_info` table did not contain a new row as expected following a node restart. (Bug #75825, Bug #20504971)

- The output format of `SHOW CREATE TABLE` for an `NDB` table containing foreign key constraints did not match that for the equivalent `InnoDB` table, which could lead to issues with some third-party applications. (Bug #75515, Bug #20364309)

- An `ALTER TABLE` statement containing comments and a partitioning option against an `NDB` table caused the SQL node on which it was executed to fail. (Bug #74022, Bug #19667566)

- **NDB Cluster APIs:** When a transaction is started from a cluster connection, `Table` and `Index` schema objects may be passed to this transaction for use. If these schema objects have been acquired from a different connection (`Ndb_cluster_connection` object), they can be deleted at any point by the deletion or disconnection of the owning connection. This can leave a connection with invalid schema objects, which causes an NDB API application to fail when these are dereferenced.

  To avoid this problem, if your application uses multiple connections, you can now set a check to detect sharing of schema objects between connections when passing a schema object to a transaction, using the `NdbTransaction::setSchemaObjectOwnerChecks()` method added in this release. When this check is enabled, the schema objects having the same names are acquired from the connection and compared to the schema objects passed to the transaction. Failure to match causes the application to fail with an error. (Bug #19785977)

- **NDB Cluster APIs:** The increase in the default number of hashmap buckets (`DefaultHashMapSize` API node configuration parameter) from 240 to 3480 in MySQL Cluster NDB 7.2.11 increased the size of the internal `DictHashMapInfo::HashMap` type considerably. This type was allocated on the stack in some `getTable()` calls which could lead to stack overflow issues for NDB API users.

  To avoid this problem, the hashmap is now dynamically allocated from the heap. (Bug #19306793)

Changes in MySQL NDB Cluster 7.4.3 (5.6.22-ndb-7.4.3)

Functionality Added or Changed

- **Important Change; NDB Cluster APIs:** This release introduces an epoch-driven Event API for the NDB API that supercedes the earlier GCI-based model. The new version of this API also simplifies error detection and handling, and monitoring of event buffer memory usage has been been improved.

  New event handling methods for `Ndb` and `NdbEventOperation` added by this change include `NdbEventOperation::getEventType2()`, `pollEvents2()`, `nextEvent2()`, `getHighestQueuedEpoch()`, `getNextEventOpInEpoch2()`, `getEpoch()`, `isEmptyEpoch()`, and `isErrorEpoch`. The `pollEvents()`, `nextEvent()`, `getLatestGCI()`, `getGCIEventOperations()`, `isConsistent()`, `isConsistentGCI()`, `getEventType()`, `getGCI()`, `getLatestGCI()`, `isOverrun()`, `hasError()`, and `clearError()` methods are deprecated beginning with the same release.

  Some (but not all) of the new methods act as replacements replacements for deprecated methods; not all of the deprecated methods map to new ones. The Event Class, provides information as to which old methods correspond to new ones.

  Error handling using the new API is no longer handled using dedicated `hasError()` and `clearError()` methods, which are now deprecated as previously noted. To support this change,

`TableEvent` now supports the values `TE_EMPTY` (empty epoch), `TE_INCONSISTENT` (inconsistent epoch), and `TE_OUT_OF_MEMORY` (insufficient event buffer memory).

Event buffer memory management has also been improved with the introduction of the `get_eventbuffer_free_percent()`, `set_eventbuffer_free_percent()`, and `get_event_buffer_memory_usage()` methods, as well as a new NDB API error `Free percent out of range` (error code 4123). Memory buffer usage can now be represented in applications using the `EventBufferMemoryUsage` data structure, and checked from MySQL client applications by reading the `ndb_eventbuffer_free_percent` system variable.

For more information, see the detailed descriptions for the `Ndb` and `NdbEventOperation` methods listed. See also Event::TableEvent, as well as The EventBufferMemoryUsage Structure.

- Additional logging is now performed of internal states occurring during system restarts such as waiting for node ID allocation and master takeover of global and local checkpoints. (Bug #74316, Bug #19795029)

- Added the `operations_per_fragment` table to the `ndbinfo` information database. Using this table, you can now obtain counts of operations performed on a given fragment (or fragment replica). Such operations include reads, writes, updates, and deletes, scan and index operations performed while executing them, and operations refused, as well as information relating to rows scanned on and returned from a given fragment replica. This table also provides information about interpreted programs used as attribute values, and values returned by them.

- Added the `MaxParallelCopyInstances` data node configuration parameter. In cases where the parallelism used during restart copy phase (normally the number of LDMs up to a maximum of 16) is excessive and leads to system overload, this parameter can be used to override the default behavior by reducing the degree of parallelism employed.

- **NDB Cluster APIs:** Two new example programs, demonstrating reads and writes of `CHAR`, `VARCHAR`, and `VARBINARY` column values, have been added to `storage/ndb/ndbapi-examples` in the MySQL Cluster source tree. For more information about these programs, including source code listings, see NDB API Simple Array Example, and NDB API Simple Array Example Using Adapter.

Bugs Fixed

- The global checkpoint commit and save protocols can be delayed by various causes, including slow disk I/O. The `DIH` master node monitors the progress of both of these protocols, and can enforce a maximum lag time during which the protocols are stalled by killing the node responsible for the lag when it reaches this maximum. This `DIH` master GCP monitor mechanism did not perform its task more than once per master node; that is, it failed to continue monitoring after detecting and handling a GCP stop. (Bug #20128256)

  References: See also: Bug #19858151, Bug #20069617, Bug #20062754.

- When running `mysql_upgrade` on a MySQL Cluster SQL node, the expected drop of the `performance_schema` database on this node was instead performed on all SQL nodes connected to the cluster. (Bug #20032861)

- The warning shown when an `ALTER TABLE ALGORITHM=INPLACE ... ADD COLUMN` statement automatically changes a column's `COLUMN_FORMAT` from `FIXED` to `DYNAMIC` now includes the name of the column whose format was changed. (Bug #20009152, Bug #74795)

- The local checkpoint scan fragment watchdog and the global checkpoint monitor can each exclude a node when it is too slow when participating in their respective protocols. This exclusion was implemented by simply asking the failing node to shut down, which in case this was delayed (for whatever reason) could prolong the duration of the GCP or LCP stall for other, unaffected nodes.

  To minimize this time, an isolation mechanism has been added to both protocols whereby any other live nodes forcibly disconnect the failing node after a predetermined amount of time. This allows the

failing node the opportunity to shut down gracefully (after logging debugging and other information) if possible, but limits the time that other nodes must wait for this to occur. Now, once the remaining live nodes have processed the disconnection of any failing nodes, they can commence failure handling and restart the related protocol or protocol, even if the failed node takes an excessively long time to shut down. (Bug #19858151)

References: See also: Bug #20128256, Bug #20069617, Bug #20062754.

- The matrix of values used for thread configuration when applying the setting of the `MaxNoOfExecutionThreads` configuration parameter has been improved to align with support for greater numbers of LDM threads. See Multi-Threading Configuration Parameters (ndbmtd), for more information about the changes. (Bug #75220, Bug #20215689)

- When a new node failed after connecting to the president but not to any other live node, then reconnected and started again, a live node that did not see the original connection retained old state information. This caused the live node to send redundant signals to the president, causing it to fail. (Bug #75218, Bug #20215395)

- In the `NDB` kernel, it was possible for a `TransporterFacade` object to reset a buffer while the data contained by the buffer was being sent, which could lead to a race condition. (Bug #75041, Bug #20112981)

- `mysql_upgrade` failed to drop and recreate the `ndbinfo` database and its tables as expected. (Bug #74863, Bug #20031425)

- Due to a lack of memory barriers, MySQL Cluster programs such as `ndbmtd` did not compile on `POWER` platforms. (Bug #74782, Bug #20007248)

- In spite of the presence of a number of protection mechanisms against overloading signal buffers, it was still in some cases possible to do so. This fix adds block-level support in the `NDB` kernel (in `SimulatedBlock`) to make signal buffer overload protection more reliable than when implementing such protection on a case-by-case basis. (Bug #74639, Bug #19928269)

- Copying of metadata during local checkpoints caused node restart times to be highly variable which could make it difficult to diagnose problems with restarts. The fix for this issue introduces signals (including `PAUSE_LCP_IDLE`, `PAUSE_LCP_REQUESTED`, and `PAUSE_NOT_IN_LCP_COPY_META_DATA`) to pause LCP execution and flush LCP reports, making it possible to block LCP reporting at times when LCPs during restarts become stalled in this fashion. (Bug #74594, Bug #19898269)

- When a data node was restarted from its angel process (that is, following a node failure), it could be allocated a new node ID before failure handling was actually completed for the failed node. (Bug #74564, Bug #19891507)

- In `NDB` version 7.4, node failure handling can require completing checkpoints on up to 64 fragments. (This checkpointing is performed by the `DBLQH` kernel block.) The requirement for master takeover to wait for completion of all such checkpoints led in such cases to excessive length of time for completion.

  To address these issues, the `DBLQH` kernel block can now report that it is ready for master takeover before it has completed any ongoing fragment checkpoints, and can continue processing these while the system completes the master takeover. (Bug #74320, Bug #19795217)

- Local checkpoints were sometimes started earlier than necessary during node restarts, while the node was still waiting for copying of the data distribution and data dictionary to complete. (Bug #74319, Bug #19795152)

- The check to determine when a node was restarting and so know when to accelerate local checkpoints sometimes reported a false positive. (Bug #74318, Bug #19795108)

- Values in different columns of the `ndbinfo` tables `disk_write_speed_aggregate` and `disk_write_speed_aggregate_node` were reported using differing multiples of bytes. Now all of these columns display values in bytes.

  In addition, this fix corrects an error made when calculating the standard deviations used in the `std_dev_backup_lcp_speed_last_10sec`, `std_dev_redo_speed_last_10sec`, `std_dev_backup_lcp_speed_last_60sec`, and `std_dev_redo_speed_last_60sec` columns of the `ndbinfo.disk_write_speed_aggregate` table. (Bug #74317, Bug #19795072)

- Recursion in the internal method `Dblqh::finishScanrec()` led to an attempt to create two list iterators with the same head. This regression was introduced during work done to optimize scans for version 7.4 of the `NDB` storage engine. (Bug #73667, Bug #19480197)

- Transporter send buffers were not updated properly following a failed send. (Bug #45043, Bug #20113145)

- **NDB Disk Data:** An update on many rows of a large Disk Data table could in some rare cases lead to node failure. In the event that such problems are observed with very large transactions on Disk Data tables you can now increase the number of page entries allocated for disk page buffer memory by raising the value of the `DiskPageBufferEntries` data node configuration parameter added in this release. (Bug #19958804)

- **NDB Disk Data:** In some cases, during `DICT` master takeover, the new master could crash while attempting to roll forward an ongoing schema transaction. (Bug #19875663, Bug #74510)

- **NDB Cluster APIs:** It was possible to delete an `Ndb_cluster_connection` object while there remained instances of `Ndb` using references to it. Now the `Ndb_cluster_connection` destructor waits for all related `Ndb` objects to be released before completing. (Bug #19999242)

  References: See also: Bug #19846392.

Changes in MySQL NDB Cluster 7.4.2 (5.6.21-ndb-7.4.2)

Functionality Added or Changed

- Added the `restart_info` table to the `ndbinfo` information database to provide current status and timing information relating to node and system restarts. By querying this table, you can observe the progress of restarts in real time. (Bug #19795152)

- After adding new data nodes to the configuration file of a MySQL Cluster having many API nodes, but prior to starting any of the data node processes, API nodes tried to connect to these "missing" data nodes several times per second, placing extra loads on management nodes and the network. To reduce unnecessary traffic caused in this way, it is now possible to control the amount of time that an API node waits between attempts to connect to data nodes which fail to respond; this is implemented in two new API node configuration parameters `StartConnectBackoffMaxTime` and `ConnectBackoffMaxTime`.

  Time elapsed during node connection attempts is not taken into account when applying these parameters, both of which are given in milliseconds with approximately 100 ms resolution. As long as the API node is not connected to any data nodes as described previously, the value of the `StartConnectBackoffMaxTime` parameter is applied; otherwise, `ConnectBackoffMaxTime` is used.

  In a MySQL Cluster with many unstarted data nodes, the values of these parameters can be raised to circumvent connection attempts to data nodes which have not yet begun to function in the cluster, as well as moderate high traffic to management nodes.

  For more information about the behavior of these parameters, see Defining SQL and Other API Nodes in an NDB Cluster. (Bug #17257842)

Bugs Fixed

- When performing a batched update, where one or more successful write operations from the start of the batch were followed by write operations which failed without being aborted (due to the `AbortOption` being set to `AO_IgnoreError`), the failure handling for these by the transaction coordinator leaked `CommitAckMarker` resources. (Bug #19875710)

  References: This issue is a regression of: Bug #19451060, Bug #73339.

- Online downgrades to MySQL Cluster NDB 7.3 failed when a MySQL Cluster NDB 7.4 master attempted to request a local checkpoint with 32 fragments from a data node already running NDB 7.3, which supports only 2 fragments for LCPs. Now in such cases, the NDB 7.4 master determines how many fragments the data node can handle before making the request. (Bug #19600834)

- The fix for a previous issue with the handling of multiple node failures required determining the number of TC instances the failed node was running, then taking them over. The mechanism to determine this number sometimes provided an invalid result which caused the number of TC instances in the failed node to be set to an excessively high value. This in turn caused redundant takeover attempts, which wasted time and had a negative impact on the processing of other node failures and of global checkpoints. (Bug #19193927)

  References: This issue is a regression of: Bug #18069334.

- The server side of an NDB transporter disconnected an incoming client connection very quickly during the handshake phase if the node at the server end was not yet ready to receive connections from the other node. This led to problems when the client immediately attempted once again to connect to the server socket, only to be disconnected again, and so on in a repeating loop, until it suceeded. Since each client connection attempt left behind a socket in `TIME_WAIT`, the number of sockets in `TIME_WAIT` increased rapidly, leading in turn to problems with the node on the server side of the transporter.

  Further analysis of the problem and code showed that the root of the problem lay in the handshake portion of the transporter connection protocol. To keep the issue described previously from occurring, the node at the server end now sends back a `WAIT` message instead of disconnecting the socket when the node is not yet ready to accept a handshake. This means that the client end should no longer need to create a new socket for the next retry, but can instead begin immediately with a new handshake hello message. (Bug #17257842)

- Corrupted messages to data nodes sometimes went undetected, causing a bad signal to be delivered to a block which aborted the data node. This failure in combination with disconnecting nodes could in turn cause the entire cluster to shut down.

  To keep this from happening, additional checks are now made when unpacking signals received over TCP, including checks for byte order, compression flag (which must not be used), and the length of the next message in the receive buffer (if there is one).

  Whenever two consecutive unpacked messages fail the checks just described, the current message is assumed to be corrupted. In this case, the transporter is marked as having bad data and no more unpacking of messages occurs until the transporter is reconnected. In addition, an entry is written to the cluster log containing the error as well as a hex dump of the corrupted message. (Bug #73843, Bug #19582925)

- During restore operations, an attribute's maximum length was used when reading variable-length attributes from the receive buffer instead of the attribute's actual length. (Bug #73312, Bug #19236945)

Changes in MySQL NDB Cluster 7.4.1 (5.6.20-ndb-7.4.1)

Node Restart Performance and Reporting Enhancements

- **Performance:** A number of performance and other improvements have been made with regard to node starts and restarts. The following list contains a brief description of each of these changes:

- Before memory allocated on startup can be used, it must be touched, causing the operating system to allocate the actual physical memory needed. The process of touching each page of memory that was allocated has now been multithreaded, with touch times on the order of 3 times shorter than with a single thread when performed by 16 threads.

- When performing a node or system restart, it is necessary to restore local checkpoints for the fragments. This process previously used delayed signals at a point which was found to be critical to performance; these have now been replaced with normal (undelayed) signals, which should shorten significantly the time required to back up a MySQL Cluster or to restore it from backup.

- Previously, there could be at most 2 LDM instances active with local checkpoints at any given time. Now, up to 16 LDMs can be used for performing this task, which increases utilization of available CPU power, and can speed up LCPs by a factor of 10, which in turn can greatly improve restart times.

  Better reporting of disk writes and increased control over these also make up a large part of this work. New `ndbinfo` tables `disk_write_speed_base`, `disk_write_speed_aggregate`, and `disk_write_speed_aggregate_node` provide information about the speed of disk writes for each LDM thread that is in use. The `DiskCheckpointSpeed` and `DiskCheckpointSpeedInRestart` configuration parameters have been deprecated, and are subject to removal in a future MySQL Cluster release. This release adds the data node configuration parameters `MinDiskWriteSpeed`, `MaxDiskWriteSpeed`, `MaxDiskWriteSpeedOtherNodeRestart`, and `MaxDiskWriteSpeedOwnRestart` to control write speeds for LCPs and backups when the present node, another node, or no node is currently restarting.

  For more information, see the descriptions of the `ndbinfo` tables and MySQL Cluster configuration parameters named previously.

- Reporting of MySQL Cluster start phases has been improved, with more frequent printouts. New and better information about the start phases and their implementation has also been provided in the sources and documentation. See Summary of NDB Cluster Start Phases.

Improved Scan and SQL Processing

- **Performance:** Several internal methods relating to the `NDB` receive thread have been optimized to make `mysqld` more efficient in processing SQL applications with the `NDB` storage engine. In particular, this work improves the performance of the `NdbReceiver::execTRANSID_AI()` method, which is commonly used to receive a record from the data nodes as part of a scan operation. (Since the receiver thread sometimes has to process millions of received records per second, it is critical that this method does not perform unnecessary work, or tie up resources that are not strictly needed.) The associated internal functions `receive_ndb_packed_record()` and `handleReceivedSignal()` methods have also been improved, and made more efficient.

Per-Fragment Memory Reporting

- Information about memory usage by individual fragments can now be obtained from the `memory_per_fragment` view added in this release to the `ndbinfo` information database. This information includes pages having fixed, and variable element size, rows, fixed element free slots, variable element free bytes, and hash index memory usage. For information, see The ndbinfo memory_per_fragment Table.

Bugs Fixed

- In some cases, transporter receive buffers were reset by one thread while being read by another. This happened when a race condition occurred between a thread receiving data and another thread initiating disconnect of the transporter (disconnection clears this buffer). Concurrency logic has now been implemented to keep this race from taking place. (Bug #19552283, Bug #73790)

- When a new data node started, API nodes were allowed to attempt to register themselves with the data node for executing transactions before the data node was ready. This forced the API node to wait an extra heartbeat interval before trying again.

  To address this issue, a number of `HA_ERR_NO_CONNECTION` errors (Error 4009) that could be issued during this time have been changed to `Cluster temporarily unavailable` errors (Error 4035), which should allow API nodes to use new data nodes more quickly than before. As part of this fix, some errors which were incorrectly categorised have been moved into the correct categories, and some errors which are no longer used have been removed. (Bug #19524096, Bug #73758)

- Executing `ALTER TABLE ... REORGANIZE PARTITION` after increasing the number of data nodes in the cluster from 4 to 16 led to a crash of the data nodes. This issue was shown to be a regression caused by previous fix which added a new dump handler using a dump code that was already in use (7019), which caused the command to execute two different handlers with different semantics. The new handler was assigned a new `DUMP` code (7024). (Bug #18550318)

  References: This issue is a regression of: Bug #14220269.

- When certain queries generated signals having more than 18 data words prior to a node failure, such signals were not written correctly in the trace file. (Bug #18419554)

- Failure of multiple nodes while using `ndbmtd` with multiple TC threads was not handled gracefully under a moderate amount of traffic, which could in some cases lead to an unplanned shutdown of the cluster. (Bug #18069334)

- For multithreaded data nodes, some threads do communicate often, with the result that very old signals can remain at the top of the signal buffers. When performing a thread trace, the signal dumper calculated the latest signal ID from what it found in the signal buffers, which meant that these old signals could be erroneously counted as the newest ones. Now the signal ID counter is kept as part of the thread state, and it is this value that is used when dumping signals for trace files. (Bug #73842, Bug #19582807)

- **NDB Cluster APIs:** When an NDB API client application received a signal with an invalid block or signal number, `NDB` provided only a very brief error message that did not accurately convey the nature of the problem. Now in such cases, appropriate printouts are provided when a bad signal or message is detected. In addition, the message length is now checked to make certain that it matches the size of the embedded signal. (Bug #18426180)