MySQL NDB Cluster 7.2 Release Notes

Abstract

This document contains release notes for the changes in each release of MySQL NDB Cluster that uses version 7.2 of the `NDB` (`NDBCLUSTER`) storage engine.

Each NDB Cluster 7.2 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 7.2.

For general information about features added in NDB Cluster 7.2, see What is New in MySQL NDB Cluster 7.2. For a complete list of all bug fixes and feature changes in MySQL Cluster, please refer to the changelog section for each individual NDB Cluster release.

For additional MySQL 5.5 documentation, see the MySQL 5.5 Reference Manual, which includes an overview of features added in MySQL 5.5 that are not specific to NDB Cluster (What Is New in MySQL 5.5), and discussion of upgrade issues that you may encounter for upgrades from MySQL 5.1 to MySQL 5.5 (Changes in MySQL 5.5). For a complete list of all bug fixes and feature changes made in MySQL 5.5 that are not specific to `NDB`, see MySQL 5.5 Release Notes.

Updates to these notes occur as new product features are added, so that everybody can follow the development process. If a recent version is listed here that you cannot find on the download page (https://dev.mysql.com/downloads/), the version has not yet been released.

The documentation included in source and binary distributions may not be fully up to date with respect to release note entries because integration of the documentation occurs at release build time. For the most up-to-date release notes, please refer to the online documentation instead.

For legal information, see the Legal Notices.

For help with using MySQL, please visit the MySQL Forums, where you can discuss your issues with other MySQL users.

Document generated on: 2020-02-03 (revision: 19716)

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 7.2 of the NDB storage engine.

Legal Notices

Documentation Accessibility

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

Access to Oracle Support

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

Changes in MySQL NDB Cluster 7.2.37 (5.5.64-ndb-7.2.37) (Not released, General availability)

MySQL NDB Cluster 7.2.37 is a new release of NDB Cluster, incorporating new features in the NDB storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.64.

Version 5.5.64-ndb-7.2.37 has no release notes, or they have not been published because the product version has not been released.

Changes in MySQL NDB Cluster 7.2.36 (5.5.63-ndb-7.2.36) (Not released, General availability)

MySQL NDB Cluster 7.2.36 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.63.

Version 5.5.63-ndb-7.2.36 has no release notes, or they have not been published because the product version has not been released.

Changes in MySQL NDB Cluster 7.2.35 (5.5.62-ndb-7.2.35) (2018-10-23, General availability)

MySQL NDB Cluster 7.2.35 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.62 (see Changes in MySQL 5.5.62 (2018-10-22, General availability)).

Bugs Fixed

- **Packaging:** The `MySQL-Cluster-devel-gpl` and `MySQL-devel` RPM packages for NDB Cluster did not provide `mysql-devel`. (Bug #66914, Bug #23525339)

Changes in MySQL NDB Cluster 7.2.34 (5.5.61-ndb-7.2.34) (2018-07-27, General availability)

MySQL NDB Cluster 7.2.34 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.61 (see Changes in MySQL 5.5.61 (2018-07-27, General availability)).

Bugs Fixed

- An internal buffer being reused immediately after it had been freed could lead to an unplanned data node shutdown. (Bug #27622643)

  References: See also: Bug #28698831.

Changes in MySQL NDB Cluster 7.2.33 (5.5.60-ndb-7.2.33) (2018-04-20, General availability)

MySQL NDB Cluster 7.2.33 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.60 (see Changes in MySQL 5.5.60 (2018-04-19, General availability)).

Bugs Fixed

- Queries using very large lists with `IN` were not handled correctly, which could lead to data node failures. (Bug #27397802)

  References: See also: Bug #28728603.

- `ndb_restore --print-data --hex` did not print trailing 0s of `LONGVARBINARY` values. (Bug #65560, Bug #14198580)

Changes in MySQL NDB Cluster 7.2.32 (5.5.59-ndb-7.2.32) (2018-01-08, General Availability)

MySQL NDB Cluster 7.2.32 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.59 (see Changes in MySQL 5.5.59 (2018-01-15, General availability)).

Bugs Fixed

- **NDB Replication:** On an SQL node not being used for a replication channel with `sql_log_bin=0` it was possible after creating and populating an NDB table for a table map event to be written to the binary log for the created table with no corresponding row events. This led to problems when this log was later used by a slave cluster replicating from the mysqld where this table was created.

  Fixed this by adding support for maintaining a cumulative `any_value` bitmap for global checkpoint event operations that represents bits set consistently for all rows of a specific table in a given epoch, and by adding a check to determine whether all operations (rows) for a specific table are all marked as `NOLOGGING`, to prevent the addition of this table to the `Table_map` held by the binlog injector.

  As part of this fix, the NDB API adds a new `getNextEventOpInEpoch3()` method which provides information about any `AnyValue` received by making it possible to retrieve the cumulative `any_value` bitmap. (Bug #26333981)

- Following `TRUNCATE TABLE` on an `NDB` table, its `AUTO_INCREMENT` ID was not reset on an SQL node not performing binary logging. (Bug #14845851)
- The `NDBFS` block's `OM_SYNC` flag is intended to make sure that all FSWRITEREQ signals used for a given file are synchronized, but was ignored by platforms that do not support `O_SYNC`, meaning that this feature did not behave properly on those platforms. Now the synchronization flag is used on those platforms that do not support `O_SYNC`. (Bug #76975, Bug #21049554)

Changes in MySQL NDB Cluster 7.2.31 (5.5.58-ndb-7.2.31) (2017-10-18, General Availability)

MySQL NDB Cluster 7.2.31 is a new release of NDB Cluster which upgrades the included MySQL Server from version 5.5.57 to 5.5.58. No changes have been made in the `NDB` storage engine for this release, which is identical in this respect with NDB 7.2.30.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

Changes in MySQL NDB Cluster 7.2.30 (5.5.57-ndb-7.2.30) (2017-07-18, General Availability)

MySQL NDB Cluster 7.2.30 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.57 (see Changes in MySQL 5.5.57 (2017-07-17, General availability)).

Functionality Added or Changed

- **Important Change; MySQL NDB ClusterJ:** The ClusterJPA plugin for OpenJPA is no longer supported by NDB Cluster, and has been removed from the distribution. (Bug #23563810)

Changes in MySQL NDB Cluster 7.2.29 (5.5.56-ndb-7.2.29) (2017-05-02, General Availability)

MySQL NDB Cluster 7.2.29 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.56 (see Changes in MySQL 5.5.56 (2017-05-02, General availability)).

Binary packages for MySQL NDB Cluster 7.2.28 are identical to those for MySQL NDB Cluster 7.2.29, except for the version number. The change in NDB 7.2.29 for Bug #25942414 is applicable only to those who build from source.

Security Notes

- For the `WITH_SSL` CMake option, `no` is no longer a permitted value or the default value. The default is now `bundled`. Consequently, the MySQL server bundled with NDB Cluster now is always built with SSL support and cannot be built without some SSL library. (Bug #25942414)

Changes in MySQL NDB Cluster 7.2.28 (5.5.55-ndb-7.2.28) (2017-04-10, General Availability)

MySQL NDB Cluster 7.2.28 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.55 (see Changes in MySQL 5.5.55 (2017-04-10, General availability)).

Bugs Fixed

- **NDB Disk Data:** Stale data from NDB Disk Data tables that had been dropped could potentially be included in backups due to the fact that disk scans were enabled for these. To prevent this possibility, disk scans are now disabled—as are other types of scans—when taking a backup. (Bug #84422, Bug #25353234)
- **NDB Disk Data:** In some cases, setting dynamic in-memory columns of an NDB Disk Data table to `NULL` was not handled correctly. (Bug #79253, Bug #22195588)

Changes in MySQL NDB Cluster 7.2.27 (5.5.54-ndb-7.2.27) (2017-01-17, General Availability)

MySQL NDB Cluster 7.2.27 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.54 (see Changes in MySQL 5.5.54 (2016-12-12, General availability)).

Bugs Fixed

- A number of potential buffer overflow issues were found and fixed in the `NDB` codebase. (Bug #25260091)

  References: See also: Bug #23152979.

- `ndb_restore` did not restore tables having more than 341 columns correctly. This was due to the fact that the buffer used to hold table metadata read from `.ctl` files was of insufficient size, so that only part

of the table descriptor could be read from it in such cases. This issue is fixed by increasing the size of the buffer used by `ndb_restore` for file reads. (Bug #25182956)

References: See also: Bug #25302901.

Changes in MySQL NDB Cluster 7.2.26 (5.5.53-ndb-7.2.26) (2016-10-18, General Availability)

MySQL NDB Cluster 7.2.26 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.53 (see Changes in MySQL 5.5.53 (2016-10-12, General Availability)).

Bugs Fixed

- 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)

Changes in MySQL NDB Cluster 7.2.25 (5.5.50-ndb-7.2.25) (2016-07-18, General Availability)

MySQL NDB Cluster 7.2.25 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.50 (see Changes in MySQL 5.5.50 (2016-06-02, General Availability)).

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)

Changes in MySQL NDB Cluster 7.2.24 (5.5.48-ndb-7.2.24) (2016-04-19, General Availability)

MySQL NDB Cluster 7.2.24 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.48 (see Changes in MySQL 5.5.48 (2016-02-05, General Availability)).

Bugs Fixed

- 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.

Changes in MySQL NDB Cluster 7.2.23 (5.5.47-ndb-7.2.23) (2016-01-19, General Availability)

MySQL NDB Cluster 7.2.23 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.47 (see Changes in MySQL 5.5.47 (2015-12-07, General Availability)).

Bugs Fixed

- **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)

- 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 NDB 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 NDB 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)

- 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)

- 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)

- `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)
- 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)

Changes in MySQL NDB Cluster 7.2.22 (5.5.46-ndb-7.2.22) (2015-10-19, General Availability)

MySQL NDB Cluster 7.2.22 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.46 (see Changes in MySQL 5.5.46 (2015-09-30, General Availability)).

Bugs Fixed

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

  References: See also: Bug #20204854, 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.

- 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)

- 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)

Changes in MySQL NDB Cluster 7.2.21 (5.5.44-ndb-7.2.21) (2015-07-13, General Availability)

MySQL NDB Cluster 7.2.21 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.44 (see Changes in MySQL 5.5.44 (2015-05-29, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- 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 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 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

- **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)

- **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)

- 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.

- The multithreaded 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.

- 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)

- 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)

Changes in MySQL NDB Cluster 7.2.20 (5.5.43-ndb-7.2.20) (2015-04-13, General Availability)

MySQL NDB Cluster 7.2.20 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.43 (see Changes in MySQL 5.5.43 (2015-04-06, 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.

- **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 NDB Cluster 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)

- **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 NDB 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 NDB Cluster 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)

- 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)

- 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)

- 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)
- 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)
- 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 7.2.19 (5.5.41-ndb-7.2.19) (2015-01-25, General Availability)

MySQL NDB Cluster 7.2.19 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.41 (see Changes in MySQL 5.5.41 (2014-11-28, General Availability)).

Bundled SSL Update (Commercial Releases)

- Starting with this release, commercial distributions of MySQL NDB Cluster 7.2 are built using OpenSSL 1.0.1i.

Functionality Added or Changed

- **MySQL NDB ClusterJ:** A new property, PROPERTY_CLUSTER_CONNECT_TIMEOUT_MGM, can now be used to set the network connection timeout. (Bug #19913569)

Bugs Fixed

- **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 Disk Data:** When a node acting as a `DICT` master fails, the arbitrator selects another node to take over in place of the failed node. During the takeover procedure, which includes cleaning up any schema transactions which are still open when the master failed, the disposition of the uncommitted schema transaction is decided. Normally this transaction be rolled back, but if it has completed a sufficient portion of a commit request, the new master finishes processing the commit. Until the fate of the transaction has been decided, no new `TRANS_END_REQ` messages from clients can be processed. In addition, since multiple concurrent schema transactions are not supported, takeover cleanup must be completed before any new transactions can be started.

  A similar restriction applies to any schema operations which are performed in the scope of an open schema transaction. The counter used to coordinate schema operation across all nodes is employed both during takeover processing and when executing any non-local schema operations. This means that starting a schema operation while its schema transaction is in the takeover phase causes this counter to be overwritten by concurrent uses, with unpredictable results.

  The scenarios just described were handled previously using a pseudo-random delay when recovering from a node failure. Now we check before the new master has rolled forward or backwards any schema transactions remaining after the failure of the previous master and avoid starting new schema transactions or performing operations using old transactions until takeover processing has cleaned up after the abandoned transaction. (Bug #19874809, Bug #74503)

- **NDB Disk Data:** When a node acting as `DICT` master fails, it is still possible to request that any open schema transaction be either committed or aborted by sending this request to the new `DICT` master. In this event, the new master takes over the schema transaction and reports back on whether the commit or abort request succeeded. In certain cases, it was possible for the new master to be misidentified—that is, the request was sent to the wrong node, which responded with an error that was interpreted by the client application as an aborted schema transaction, even in cases where the transaction could have been successfully committed, had the correct node been contacted. (Bug #74521, Bug #19880747)

- **NDB Replication:** It was possible using wildcards to set up conflict resolution for an exceptions table (that is, a table named using the suffix `$EX`), which should not be allowed. Now when a replication conflict function is defined using wildcard expressions, these are checked for possible matches so that, in the event that the function would cover an exceptions table, it is not set up for this table. (Bug #19267720)

- **NDB Cluster APIs:** The buffer allocated by an `NdbScanOperation` for receiving scanned rows was not released until the `NdbTransaction` owning the scan operation was closed. This could lead to excessive memory usage in an application where multiple scans were created within the same transaction, even if these scans were closed at the end of their lifecycle, unless `NdbScanOperation::close()` was invoked with the *`releaseOp`* argument equal to `true`. Now the buffer is released whenever the cursor navigating the result set is closed with `NdbScanOperation::close()`, regardless of the value of this argument. (Bug #75128, Bug #20166585)

- **MySQL NDB ClusterJ:** The following errors were logged at the `SEVERE` level; they are now logged at the `NORMAL` level, as they should be:

  - Duplicate primary key

  - Duplicate unique key

- Foreign key constraint error: key does not exist
- Foreign key constraint error: key exists

(Bug #20045455)

- **MySQL NDB ClusterJ:** The `com.mysql.clusterj.tie` class gave off a logging message at the `INFO` logging level for every single query, which was unnecessary and was affecting the performance of applications that used ClusterJ. (Bug #20017292)

- **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 NDB 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.

- 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.

- A number of problems relating to the fired triggers pool have been fixed, including the following issues:

  - When the fired triggers pool was exhausted, `NDB` returned Error 218 (`Out of LongMessageBuffer`). A new error code 221 is added to cover this case.
  - An additional, separate case in which Error 218 was wrongly reported now returns the correct error.
  - Setting low values for `MaxNoOfFiredTriggers` led to an error when no memory was allocated if there was only one hash bucket.
  - An aborted transaction now releases any fired trigger records it held. Previously, these records were held until its `ApiConnectRecord` was reused by another transaction.
  - In addition, for the `Fired Triggers` pool in the internal `ndbinfo.ndb$pools` table, the high value always equalled the total, due to the fact that all records were momentarily seized when initializing them. Now the high value shows the maximum following completion of initialization.

  (Bug #19976428)

- Online reorganization when using `ndbmtd` data nodes and with binary logging by `mysqld` enabled could sometimes lead to failures in the `TRIX` and `DBLQH` kernel blocks, or in silent data corruption. (Bug #19903481)

  References: See also: Bug #19912988.

- 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.

- A watchdog failure resulted from a hang while freeing a disk page in `TUP_COMMITREQ`, due to use of an uninitialized block variable. (Bug #19815044, Bug #74380)
- Multiple threads crashing led to multiple sets of trace files being printed and possibly to deadlocks. (Bug #19724313)
- When a client retried against a new master a schema transaction that failed previously against the previous master while the latter was restarting, the lock obtained by this transaction on the new master prevented the previous master from progressing past start phase 3 until the client was terminated, and resources held by it were cleaned up. (Bug #19712569, Bug #74154)
- 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)
- When executing very large pushdown joins involving one or more indexes each defined over several columns, it was possible in some cases for the `DBSPJ` block (see The DBSPJ Block) in the `NDB` kernel to generate `SCAN_FRAGREQ` signals that were excessively large. This caused data nodes to fail when these could not be handled correctly, due to a hard limit in the kernel on the size of such signals (32K). This fix bypasses that limitation by breaking up `SCAN_FRAGREQ` data that is too large for one such signal, and sending the `SCAN_FRAGREQ` as a chunked or fragmented signal instead. (Bug #19390895)
- `ndb_index_stat` sometimes failed when used against a table containing unique indexes. (Bug #18715165)
- Queries against tables containing a CHAR(0) columns failed with `ERROR 1296 (HY000): Got error 4547 'RecordSpecification has overlapping offsets' from NDBCLUSTER`. (Bug #14798022)
- `ndb_restore` failed while restoring a table which contained both a built-in conversion on the primary key and a staging conversion on a `TEXT` column.

  During staging, a `BLOB` table is created with a primary key column of the target type. However, a conversion function was not provided to convert the primary key values before loading them into the staging blob table, which resulted in corrupted primary key values in the staging `BLOB` table. While moving data from the staging table to the target table, the `BLOB` read failed because it could not find the primary key in the `BLOB` table.

  Now all `BLOB` tables are checked to see whether there are conversions on primary keys of their main tables. This check is done after all the main tables are processed, so that conversion functions and

parameters have already been set for the main tables. Any conversion functions and parameters used for the primary key in the main table are now duplicated in the `BLOB` table. (Bug #73966, Bug #19642978)

- 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)

- `ndb_restore --print-data` truncated `TEXT` and `BLOB` column values to 240 bytes rather than 256 bytes. (Bug #65467, Bug #14571512)

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

Changes in MySQL NDB Cluster 7.2.18 (5.5.40-ndb-7.2.18) (2014-10-16, General Availability)

MySQL NDB Cluster 7.2.18 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.40 (see Changes in MySQL 5.5.40 (2014-09-22, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Added the `--exclude-missing-tables` option for `ndb_restore`. When enabled, the option causes tables present in the backup but not in the target database to be ignored. (Bug #57566, Bug #11764704)

Bugs Fixed

- **NDB Cluster APIs:** The fix for Bug #16723708 stopped the `ndb_logevent_get_next()` function from casting a log event's `ndb_mgm_event_category` to an `enum` type, but this change interfered with existing applications, and so the function's original behavior is now reinstated. A new MGM API function exhibiting the corrected behavior `ndb_logevent_get_next2()` has been added in this release to take the place of the reverted function, for use in applications that do not require backward compatibility. In all other respects apart from this, the new function is identical with its predecessor. (Bug #18354165)

References: Reverted patches: Bug #16723708.

- **NDB Cluster APIs:** NDB API scans leaked `Ndb_cluster_connection` objects after `nextResult()` was called when an operation resulted in an error. This leak locked up the corresponding connection objects in the `DBTC` kernel block until the connection was closed. (Bug #17730825, Bug #20170731)

- **MySQL NDB ClusterJ:** Retrieval of values from `BLOB` and `TEXT` columns by ClusterJ column accessor methods was not handled correctly. (Bug #18419468, Bug #19028487)

- When assembling error messages of the form `Incorrect state for node` *`n`* `state:` *`node_state`*, written when the transporter failed to connect, the node state was used in place of the node ID in a number of instances, which resulted in errors of this type for which the node state was reported incorrectly. (Bug #19559313, Bug #73801)

- 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)

- The failure of a data node could in some situations cause a set of API nodes to fail as well due to the sending of a `CLOSE_COMREQ` signal that was sometimes not completely initialized. (Bug #19513967)

- A more detailed error report is printed in the event of a critical failure in one of the `NDB` internal `sendSignal*()` methods, prior to crashing the process, as was already implemented for `sendSignal()`, but was missing from the more specialized `sendSignalNoRelease()` method. Having a crash of this type correctly reported can help with identifying configuration hardware issues in some cases. (Bug #19414511)

  References: See also: Bug #19390895.

- `ndb_restore` failed to restore the cluster's metadata when there were more than approximately 17 K data objects. (Bug #19202654)

- 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.

- Parallel transactions performing reads immediately preceding a delete on the same tuple could cause the `NDB` kernel to crash. This was more likely to occur when separate TC threads were specified using the `ThreadConfig` configuration parameter. (Bug #19031389)

- Attribute promotion between different `TEXT` types (any of `TINYTEXT`, `TEXT`, `MEDIUMTEXT`, and `LONGTEXT`) by `ndb_restore` was not handled properly in some cases. In addition, `TEXT` values are now truncated according to the limits set by `mysqld` (for example, values converted to `TINYTEXT` from another type are truncated to 256 bytes). In the case of columns using a multibyte character set, the value is truncated to the end of the last well-formed character.

  Also as a result of this fix, conversion to a `TEXT` column of any size that uses a different character set from the original is now disallowed. (Bug #18875137)

- To assist with diagnostic issues where many watchdog warnings are raised, it is now possible to activate (or deactivate) a killer watchdog using `DUMP 2610` in the `ndb_mgm` client. When set, this shuts down the data node on which the next watchdog warning occurs, providing a trace log. (Bug #18703922)

- The `NDB` optimized node recovery mechanism attempts to transfer only relevant page changes to a starting node in order to speed the recovery process; this is done by having the starting node indicate the index of the last global checkpoint (GCI) in which it participated, so that the node that was already running copies only data for rows which have changed since that GCI. Every row has a GCI metacolumn which facilitates this; for a deleted row, the slot formerly stpring this row's data contains a GCI value, and for deleted pages, every row on the missing page is considered changed and thus needs to be sent.

  When these changes are received by the starting node, this node performs a lookup for the page and index to determine what they contain. This lookup could cause a real underlying page to be mapped against the logical page ID, even when this page contained no data.

  One way in which this issue could manifest itself occurred after cluster `DataMemory` usage approached maximum, and deletion of many rows followed by a rolling restart of the data nodes was performed with the expectation that this would free memory, but in fact it was possible in this scenario for memory not to be freed and in some cases for memory usage actually to increase to its maximum.

  This fix solves these issues by ensuring that a real physical page is mapped to a logical ID during node recovery only when this page contains actual data which needs to be stored. (Bug #18683398, Bug #18731008)

- `mysqld` failed while attempting a read removal before a delete with an index merge. This occurred only when a quick select was also generated for the delete operation. During read removal, the index from the quick select is used to access the table structure. In this case, because the delete uses an index merge, the quick select index is set to `MAX_KEY` instead of a valid index value, which led to a bad pointer (which was then dereferenced).

  The fix for this problem adds a check so that read removal before delete is not attempted if the quick select index has a value of `MAX_KEY`. (Bug #18487960)

- When a data node sent a `MISSING_DATA` signal due to a buffer overflow and no event data had yet been sent for the current epoch, the dummy event list created to handle this inconsistency was not deleted after the information in the dummy event list was transferred to the completed list. (Bug #18410939)

- Incorrect calculation of the next autoincrement value following a manual insertion towards the end of a cached range could result in duplicate values sometimes being used. This issue could manifest itself when using certain combinations of values for `auto_increment_increment`, `auto_increment_offset`, and `ndb_autoincrement_prefetch_sz`.

  This issue has been fixed by modifying the calculation to make sure that the next value from the cache as computed by `NDB` is of the form `auto_increment_offset + (N * auto_increment_increment`. This avoids any rounding up by the MySQL Server of the returned value, which could result in duplicate entries when the rounded-up value fell outside the range of values cached by `NDB`. (Bug #17893872)

- `ndb_show_tables --help` output contained misleading information about the `--database` (`-d`) option. In addition, the long form of the option (`--database`) did not work properly. (Bug #17703874)

- Using the `--help` option with `ndb_print_file` caused the program to segfault. (Bug #17069285)

- 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)

Changes in MySQL NDB Cluster 7.2.17 (5.5.37-ndb-7.2.17) (2014-07-11, General Availability)

MySQL NDB Cluster 7.2.17 is a new release of NDB Cluster, incorporating new features in the NDB storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.37 (see Changes in MySQL 5.5.37 (2014-03-27, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **NDB Cluster APIs:** Added as an aid to debugging the ability to specify a human-readable name for a given `Ndb` object and later to retrieve it. These operations are implemented, respectively, as the `setNdbObjectName()` and `getNdbObjectName()` methods.

  To make tracing of event handling between a user application and `NDB` easier, you can use the reference (from `getReference()` followed by the name (if provided) in printouts; the reference ties together the application `Ndb` object, the event buffer, and the `NDB` storage engine's `SUMA` block. (Bug #18419907)

Bugs Fixed

- **NDB Replication:** When using `NDB$EPOCH_TRANS`, conflicts between `DELETE` operations were handled like conflicts between updates, with the primary rejecting the transaction and dependents, and realigning the secondary. This meant that their behavior with regard to subsequent operations on any affected row or rows depended on whether they were in the same epoch or a different one: within the same epoch, they were considered conflicting events; in different epochs, they were not considered in conflict.

  This fix brings the handling of conflicts between deletes by `NDB$EPOCH_TRANS` with that performed when using `NDB$EPOCH` for conflict detection and resolution, and extends testing with `NDB$EPOCH` and `NDB$EPOCH_TRANS` to include “delete-delete” conflicts, and encapsulate the expected result, with transactional conflict handling modified so that a conflict between `DELETE` operations *alone* is not sufficient to cause a transaction to be considered in conflict. (Bug #18459944)

- **NDB Cluster APIs:** When an `NDB` data node indicates a buffer overflow via an empty epoch, the event buffer places an inconsistent data event in the event queue. When this was consumed, it was not removed from the event queue as expected, causing subsequent `nextEvent()` calls to return 0. This caused event consumption to stall because the inconsistency remained flagged forever, while event data accumulated in the queue.

  Event data belonging to an empty inconsistent epoch can be found either at the beginning or somewhere in the middle. `pollEvents()` returns 0 for the first case. This fix handles the second case: calling `nextEvent()` call dequeues the inconsistent event before it returns. In order to benefit from this fix, user applications must call `nextEvent()` even when `pollEvents()` returns 0. (Bug #18716991)

- **NDB Cluster APIs:** The `pollEvents()` method returned 1, even when called with a wait time equal to 0, and there were no events waiting in the queue. Now in such cases it returns 0 as expected. (Bug #18703871)

- **MySQL NDB ClusterJ:** Writing a value failed when read from a fixed-width `char` column using `utf8` to another column of the same type and length but using `latin1`. The data was returned with extra spaces after being padded during its insertion. The value is now trimmed before returning it.

  This fix also corrects `Data length too long` errors during the insertion of valid `utf8` characters of 2 or more bytes. This was due to padding of the data before encoding it, rather than after. (Bug #71435, Bug #18283369)

- Processing a NODE_FAILREP signal that contained an invalid node ID could cause a data node to fail. (Bug #18993037, Bug #73015)

  References: This issue is a regression of: Bug #16007980.

- When building out of source, some files were written to the source directory instead of the build dir. These included the `manifest.mf` files used for creating ClusterJ jars and the `pom.xml` file used by `mvn_install_ndbjtie.sh`. In addition, `ndbinfo.sql` was written to the build directory, but marked as output to the source directory in `CMakeLists.txt`. (Bug #18889568, Bug #72843)

- It was possible for a data node restart to become stuck indefinitely in start phase 101 (see Summary of NDB Cluster Start Phases) when there were connection problems between the node being restarted and one or more subscribing API nodes.

  To help prevent this from happening, a new data node configuration parameter `RestartSubscriberConnectTimeout` has been introduced, which can be used to control how long a data node restart can stall in start phase 101 before giving up and attempting to restart again. The default is 12000 ms. (Bug #18599198)

- 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.

- Following a long series of inserts, when running with a relatively small redo log and an insufficient large value for `MaxNoOfConcurrentTransactions`, there remained transactions that were blocked by the lack of redo log and were thus not aborted in the correct state (waiting for prepare log to be sent to disk, or `LOG_QUEUED` state). This caused the redo log to remain blocked until unblocked by a completion of a local checkpoint. This could lead to a deadlock, when the blocked aborts in turned blocked global checkpoints, and blocked GCPs block LCPs. To prevent this situation from arising, we now abort immediately when we reach the `LOG_QUEUED` state in the abort state handler. (Bug #18533982)

- `ndbmtd` supports multiple parallel receiver threads, each of which performs signal reception for a subset of the remote node connections (transporters) with the mapping of remote_nodes to receiver threads

decided at node startup. Connection control is managed by the multi-instance TRPMAN block, which is organized as a proxy and workers, and each receiver thread has a TRPMAN worker running locally.

The QMGR block sends signals to TRPMAN to enable and disable communications with remote nodes. These signals are sent to the TRPMAN proxy, which forwards them to the workers. The workers themselves decide whether to act on signals, based on the set of remote nodes they manage.

The current issue arises because the mechanism used by the TRPMAN workers for determining which connections they are responsible for was implemented in such a way that each worker thought it was responsible for all connections. This resulted in the TRPMAN actions for OPEN_COMORD, ENABLE_COMREQ, and CLOSE_COMREQ being processed multiple times.

The fix keeps TRPMAN instances (receiver threads) executing OPEN_COMORD, ENABLE_COMREQ and CLOSE_COMREQ requests. In addition, the correct TRPMAN instance is now chosen when routing from this instance for a specific remote connection. (Bug #18518037)

- During data node failure handling, the transaction coordinator performing takeover gathers all known state information for any failed TC instance transactions, determines whether each transaction has been committed or aborted, and informs any involved API nodes so that they can report this accurately to their clients. The TC instance provides this information by sending TCKEY_FAILREF or TCKEY_FAILCONF signals to the API nodes as appropriate top each affected transaction.

  In the event that this TC instance does not have a direct connection to the API node, it attempts to deliver the signal by routing it through another data node in the same node group as the failing TC, and sends a GSN_TCKEY_FAILREFCONF_R signal to TC block instance 0 in that data node. A problem arose in the case of multiple transaction cooridnators, when this TC instance did not have a signal handler for such signals, which led it to fail.

  This issue has been corrected by adding a handler to the TC proxy block which in such cases forwards the signal to one of the local TC worker instances, which in turn attempts to forward the signal on to the API node. (Bug #18455971)

- When running with a very slow main thread, and one or more transaction coordinator threads, on different CPUs, it was possible to encounter a timeout when sending a DIH_SCAN_GET_NODESREQ signal, which could lead to a crash of the data node. Now in such cases the timeout is avoided. (Bug #18449222)

- 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)

- A local checkpoint (LCP) is tracked using a global LCP state (c_lcpState), and each NDB table has a status indicator which indicates the LCP status of that table (tabLcpStatus). If the global LCP state is LCP_STATUS_IDLE, then all the tables should have an LCP status of TLS_COMPLETED.

  When an LCP starts, the global LCP status is LCP_INIT_TABLES and the thread starts setting all the NDB tables to TLS_ACTIVE. If any tables are not ready for LCP, the LCP initialization

procedure continues with CONTINUEB signals until all tables have become available and been marked TLS_ACTIVE. When this initialization is complete, the global LCP status is set to LCP_STATUS_ACTIVE.

This bug occurred when the following conditions were met:

- An LCP was in the LCP_INIT_TABLES state, and some but not all tables had been set to TLS_ACTIVE.
- The master node failed before the global LCP state changed to LCP_STATUS_ACTIVE; that is, before the LCP could finish processing all tables.
- The NODE_FAILREP signal resulting from the node failure was processed before the final CONTINUEB signal from the LCP initialization process, so that the node failure was processed while the LCP remained in the LCP_INIT_TABLES state.

Following master node failure and selection of a new one, the new master queries the remaining nodes with a MASTER_LCPREQ signal to determine the state of the LCP. At this point, since the LCP status was LCP_INIT_TABLES, the LCP status was reset to LCP_STATUS_IDLE. However, the LCP status of the tables was not modified, so there remained tables with TLS_ACTIVE. Afterwards, the failed node is removed from the LCP. If the LCP status of a given table is TLS_ACTIVE, there is a check that the global LCP status is not LCP_STATUS_IDLE; this check failed and caused the data node to fail.

Now the MASTER_LCPREQ handler ensures that the tabLcpStatus for all tables is updated to TLS_COMPLETED when the global LCP status is changed to LCP_STATUS_IDLE. (Bug #18044717)

- When performing a copying ALTER TABLE operation, mysqld creates a new copy of the table to be altered. This intermediate table, which is given a name bearing the prefix #sql-, has an updated schema but contains no data. mysqld then copies the data from the original table to this intermediate table, drops the original table, and finally renames the intermediate table with the name of the original table.

  mysqld regards such a table as a temporary table and does not include it in the output from SHOW TABLES; mysqldump also ignores an intermediate table. However, NDB sees no difference between such an intermediate table and any other table. This difference in how intermediate tables are viewed by mysqld (and MySQL client programs) and by the NDB storage engine can give rise to problems when performing a backup and restore if an intermediate table existed in NDB, possibly left over from a failed ALTER TABLE that used copying. If a schema backup is performed using mysqldump and the mysql client, this table is not included. However, in the case where a data backup was done using the ndb_mgm client's BACKUP command, the intermediate table was included, and was also included by ndb_restore, which then failed due to attempting to load data into a table which was not defined in the backed up schema.

  To prevent such failures from occurring, ndb_restore now by default ignores intermediate tables created during ALTER TABLE operations (that is, tables whose names begin with the prefix #sql-). A new option --exclude-intermediate-sql-tables is added that makes it possible to override the new behavior. The option's default value is TRUE; to cause ndb_restore to revert to the old behavior and to attempt to restore intermediate tables, set this option to FALSE. (Bug #17882305)

- The logging of insert failures has been improved. This is intended to help diagnose occasional issues seen when writing to the mysql.ndb_binlog_index table. (Bug #17461625)

- Employing a CHAR column that used the UTF8 character set as a table's primary key column led to node failure when restarting data nodes. Attempting to restore a table with such a primary key also caused ndb_restore to fail. (Bug #16895311, Bug #68893)

- The `--order` (`-o`) option for the `ndb_select_all` utility worked only when specified as the last option, and did not work with an equals sign.

  As part of this fix, the program's `--help` output was also aligned with the `--order` option's correct behavior. (Bug #64426, Bug #16374870)

Changes in MySQL NDB Cluster 7.2.16 (5.5.37-ndb-7.2.16) (2014-04-08, General Availability)

MySQL NDB Cluster 7.2.16 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.37 (see Changes in MySQL 5.5.37 (2014-03-27, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Handling of `LongMessageBuffer` shortages and statistics has been improved as follows:
  - The default value of `LongMessageBuffer` has been increased from 4 MB to 64 MB.
  - When this resource is exhausted, a suitable informative message is now printed in the data node log describing possible causes of the problem and suggesting possible solutions.
  - `LongMessageBuffer` usage information is now shown in the `ndbinfo.memoryusage` table. See the description of this table for an example and additional information.

Bugs Fixed

- **Important Change:** The server system variables `ndb_index_cache_entries` and `ndb_index_stat_freq`, which had been deprecated in a previous MySQL NDB Cluster release series, have now been removed. (Bug #11746486, Bug #26673)
- **NDB Replication:** A slave in MySQL NDB Cluster Replication now monitors the progression of epoch numbers received from its immediate upstream master, which can both serve as a useful check on the low-level functioning of replication, and provide a warning in the event replication is restarted accidentally at an already-applied position.

  As a result of this enhancement, an epoch ID collision has the following results, depending on the state of the slave SQL thread:
  - Following a `RESET SLAVE` statement, no action is taken, in order to allow the execution of this statement without spurious warnings.
  - Following `START SLAVE`, a warning is produced that the slave is being positioned at an epoch that has already been applied.

- In all other cases, the slave SQL thread is stopped against the possibility that a system malfunction has resulted in the re-application of an existing epoch.

(Bug #17461576)

References: See also: Bug #17369118.

- **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)

- **NDB Cluster APIs:** Refactoring that was performed in MySQL NDB Cluster 7.2.15 inadvertently introduced a dependency in `Ndb.hpp` on a file that is not included in the distribution, which caused NDB API applications to fail to compile. The dependency has been removed. (Bug #18293112, Bug #71803)

  References: This issue is a regression of: Bug #17647637.

- **NDB Cluster APIs:** An NDB API application sends a scan query to a data node; the scan is processed by the transaction coordinator (TC). The TC forwards a `LQHKEYREQ` request to the appropriate LDM, and aborts the transaction if it does not receive a `LQHKEYCONF` response within the specified time limit. After the transaction is successfully aborted, the TC sends a `TCROLLBACKREP` to the NDBAPI client, and the NDB API client processes this message by cleaning up any `Ndb` objects associated with the transaction.

  The client receives the data which it has requested in the form of `TRANSID_AI` signals, buffered for sending at the data node, and may be delivered after a delay. On receiving such a signal, `NDB` checks the transaction state and ID: if these are as expected, it processes the signal using the `Ndb` objects associated with that transaction.

  The current bug occurs when all the following conditions are fulfilled:

  - The transaction coordinator aborts a transaction due to delays and sends a `TCROLLBACPREP` signal to the client, while at the same time a `TRANSID_AI` which has been buffered for delivery at an LDM is delivered to the same client.
  - The NDB API client considers the transaction complete on receipt of a `TCROLLBACKREP` signal, and immediately closes the transaction.
  - The client has a separate receiver thread running concurrently with the thread that is engaged in closing the transaction.
  - The arrival of the late `TRANSID_AI` interleaves with the closing of the user thread's transaction such that `TRANSID_AI` processing passes normal checks before `closeTransaction()` resets the transaction state and invalidates the receiver.

  When these conditions are all met, the receiver thread proceeds to continue working on the `TRANSID_AI` signal using the invalidated receiver. Since the receiver is already invalidated, its usage results in a node failure.

  Now the `Ndb` object cleanup done for `TCROLLBACKREP` includes invalidation of the transaction ID, so that, for a given transaction, any signal which is received after the `TCROLLBACKREP` arrives does not pass the transaction ID check and is silently dropped. This fix is also implemented for the `TC_COMMITREF`, `TCROLLBACKREF`, `TCKEY_FAILCONF`, and `TCKEY_FAILREF` signals as well.

  See also Operations and Signals, for additional information about NDB messaging. (Bug #18196562)

- **NDB Cluster APIs:** The example `ndbapi-examples/ndbapi_blob_ndbrecord/main.cpp` included an internal header file (`ndb_global.h`) not found in the MySQL NDB Cluster binary distribution. The example now uses `stdlib.h` and `string.h` instead of this file. (Bug #18096866, Bug #71409)

- **NDB Cluster APIs:** `ndb_restore` could sometimes report `Error 701 System busy with other schema operation` unnecessarily when restoring in parallel. (Bug #17916243)

- When an `ALTER TABLE` statement changed table schemas without causing a change in the table's partitioning, the new table definition did not copy the hash map from the old definition, but used the current default hash map instead. However, the table data was not reorganized according to the new hashmap, which made some rows inaccessible using a primary key lookup if the two hash maps had incompatible definitions.

  To keep this situation from occurring, any `ALTER TABLE` that entails a hashmap change now triggers a reorganisation of the table. In addition, when copying a table definition in such cases, the hashmap is now also copied. (Bug #18436558)

- 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)

- Checking of timeouts is handled by the signal `TIME_SIGNAL`. Previously, this signal was generated by the `QMGR` `NDB` kernel block in the main thread, and sent to the `QMRG`, `DBLQH`, and `DBTC` blocks (see NDB Kernel Blocks) as needed to check (respectively) heartbeats, disk writes, and transaction timeouts. In `ndbmtd` (as opposed to `ndbd`), these blocks all execute in different threads. This meant that if, for example, `QMGR` was actively working and some other thread was put to sleep, the previously sleeping thread received a large number of `TIME_SIGNAL` messages simultaneously when it was woken up again, with the effect that effective times moved very quickly in `DBLQH` as well as in `DBTC`. In `DBLQH`, this had no noticeable adverse effects, but this was not the case in `DBTC`; the latter block could not work on transactions even though time was still advancing, leading to a situation in which many operations appeared to time out because the transaction coordinator (TC) thread was comparatively slow in answering requests.

  In addition, when the TC thread slept for longer than 1500 milliseconds, the data node crashed due to detecting that the timeout handling loop had not yet stopped. To rectify this problem, the generation of the `TIME_SIGNAL` has been moved into the local threads instead of `QMGR`; this provides for better control over how quickly `TIME_SIGNAL` messages are allowed to arrive. (Bug #18417623)

- The `ServerPort` and `TcpBind_INADDR_ANY` configuration parameters were not included in the output of `ndb_mgmd --print-full-config`. (Bug #18366909)

- After dropping an `NDB` table, neither the cluster log nor the output of the `REPORT MemoryUsage` command showed that the `IndexMemory` used by that table had been freed, even though the memory had in fact been deallocated. This issue was introduced in MySQL NDB Cluster 7.2.14. (Bug #18296810)

- `ndb_show_tables` sometimes failed with the error message `Unable to connect to management server` and immediately terminated, without providing the underlying reason for the failure. To provide more useful information in such cases, this program now also prints the most recent error from the `Ndb_cluster_connection` object used to instantiate the connection. (Bug #18276327)

- `-DWITH_NDBMTD=0` did not function correctly, which could cause the build to fail on platforms such as ARM and Raspberry Pi which do not define the memory barrier functions required to compile `ndbmtd`. (Bug #18267919)

  References: See also: Bug #16620938.

- The block threads managed by the multithreading scheduler communicate by placing signals in an out queue or job buffer which is set up between all block threads. This queue has a fixed maximum size, such that when it is filled up, the worker thread must wait for the consumer to drain the queue. In a highly loaded system, multiple threads could end up in a circular wait lock due to full out buffers, such that they were preventing each other from performing any useful work. This condition eventually led to the data node being declared dead and killed by the watchdog timer.

  To fix this problem, we detect situations in which a circular wait lock is about to begin, and cause buffers which are otherwise held in reserve to become available for signal processing by queues which are highly loaded. (Bug #18229003)

- The `ndb_mgm` client `START BACKUP` command (see Commands in the NDB Cluster Management Client) could experience occasional random failures when a ping was received prior to an expected `BackupCompleted` event. Now the connection established by this command is not checked until it has been properly set up. (Bug #18165088)

- The local checkpoint lag watchdog tracking the number of times a check for LCP timeout was performed using the system scheduler and used this count to check for a timeout condition, but this caused a number of issues. To overcome these limitations, the LCP watchdog has been refactored to keep track of its own start times, and to calculate elapsed time by reading the (real) clock every time it is called. (Bug #17842035)

  References: See also: Bug #17647469.

- Data nodes running `ndbmtd` could stall while performing an online upgrade of a MySQL NDB Cluster containing a great many tables from a version prior to NDB 7.2.5 to version 7.2.5 or later. (Bug #16693068)

Changes in MySQL NDB Cluster 7.2.15 (5.5.35-ndb-7.2.15) (2014-02-05, General Availability)

MySQL NDB Cluster 7.2.15 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.35 (see Changes in MySQL 5.5.35 (2013-12-03, General Availability)).

Bugs Fixed

- **Packaging; Solaris:** Compilation of `ndbmtd` failed on Solaris 10 and 11 for 32-bit `x86`, and the binary was not included in the binary distributions for these platforms. (Bug #16620938)

- **Microsoft Windows:** Timers used in timing scheduler events in the `NDB` kernel have been refactored, in part to insure that they are monotonic on all platforms. In particular, on Windows, event intervals were previously calculated using values obtained from `GetSystemTimeAsFileTime()`, which reads directly from the system time ("wall clock"), and which may arbitrarily be reset backward or forward, leading to false watchdog or heartbeat alarms, or even node shutdown. Lack of timer monotonicity could also cause slow disk writes during backups and global checkpoints. To fix this issue, the Windows implementation now uses `QueryPerformanceCounters()` instead of `GetSystemTimeAsFileTime()`. In the event that a monotonic timer is not found on startup of the data nodes, a warning is logged.

In addition, on all platforms, a check is now performed at compile time for available system monotonic timers, and the build fails if one cannot be found; note that `CLOCK_HIGHRES` is now supported as an alternative for `CLOCK_MONOTONIC` if the latter is not available. (Bug #17647637)

- **NDB Disk Data:** When using Disk Data tables and `ndbmtd` data nodes, it was possible for the undo buffer to become overloaded, leading to a crash of the data nodes. This issue was more likely to be encountered when using Disk Data columns whose size was approximately 8K or larger. (Bug #16766493)

- **NDB Cluster APIs:** `UINT_MAX64` was treated as a signed value by Visual Studio 2010. To prevent this from happening, the value is now explicitly defined as unsigned. (Bug #17947674)

  References: See also: Bug #17647637.

- **NDB Cluster APIs:** It was possible for an `Ndb` object to receive signals for handling before it was initialized, leading to thread interleaving and possible data node failure when executing a call to `Ndb::init()`. To guard against this happening, a check is now made when it is starting to receive signals that the `Ndb` object is properly initialized before any signals are actually handled. (Bug #17719439)

- **NDB Cluster APIs:** Compilation of example NDB API program files failed due to missing include directives. (Bug #17672846, Bug #70759)

- **ndbmemcache:** When attempting to start memcached with a `cache_size` larger than that of the available memory and with `preallocate=true` failed, the error message provided only a numeric code, and did not indicate what the actual source of the error was. (Bug #17509293, Bug #70403)

- **ndbmemcache:** A memcached server running the NDB engine could crash after being disconnected from a cluster. (Bug #14055851)

- **MySQL NDB ClusterJ:** Call of setPartitionKey() failed after a previous transaction failed. It was because the state of the transaction was not properly cleaned up after the transaction failure. This fix adds the clean-up that is needed in the situation. (Bug #17885485)

- Monotonic timers on several platforms can experience issues which might result in the monotonic clock doing small jumps back in time. This is due to imperfect synchronization of clocks between multiple CPU cores and does not normally have an adverse effect on the scheduler and watchdog mechanisms; so we handle some of these cases by making backtick protection less strict, although we continue to ensure that the backtick is less than 10 milliseconds. This fix also removes several checks for backticks which are thereby made redundant. (Bug #17973819)

- Under certain specific circumstances, in a cluster having two SQL nodes, one of these could hang, and could not be accessed again even after killing the `mysqld` process and restarting it. (Bug #17875885, Bug #18080104)

  References: See also: Bug #17934985.

- Poor support or lack of support on some platforms for monotonic timers caused issues with delayed signal handling by the job scheduler for the multithreaded data node. Variances (timer leaps) on such platforms are now handled in the same way the multithreaded data node process that they are by the singlethreaded version. (Bug #17857442)

  References: See also: Bug #17475425, Bug #17647637.

- In some cases, with `ndb_join_pushdown` enabled, it was possible to obtain from a valid query the error `Got error 290 'Corrupt key in TC, unable to xfrm' from NDBCLUSTER` even though the data was not actually corrupted.

It was determined that a `NULL` in a `VARCHAR` column could be used to construct a lookup key, but since `NULL` is never equal to any other value, such a lookup could simple have been eliminated instead. This `NULL` lookup in turn led to the spurious error message.

This fix takes advantage of the fact that a key lookup with `NULL` never finds any matching rows, and so `NDB` does not try to perform the lookup that would have led to the error. (Bug #17845161)

- It was theoretically possible in certain cases for a number of output functions internal to the `NDB` code to supply an uninitialized buffer as output. Now in such cases, a newline character is printed instead. (Bug #17775602, Bug #17775772)

- Use of the `localtime()` function in `NDB` multithreading code led to otherwise nondeterministic failures in `ndbmtd`. This fix replaces this function, which on many platforms uses a buffer shared among multiple threads, with `localtime_r()`, which can have allocated to it a buffer of its own. (Bug #17750252)

- When using single-threaded (`ndbd`) data nodes with `RealTimeScheduler` enabled, the CPU did not, as intended, temporarily lower its scheduling priority to normal every 10 milliseconds to give other, non-realtime threads a chance to run. (Bug #17739131)

- During arbitrator selection, `QMGR` (see The QMGR Block) runs through a series of states, the first few of which are (in order) `NULL`, `INIT`, `FIND`, `PREP1`, `PREP2`, and `START`. A check for an arbitration selection timeout occurred in the `FIND` state, even though the corresponding timer was not set until `QMGR` reached the `PREP1` and `PREP2` states. Attempting to read the resulting uninitialized timestamp value could lead to false `Could not find an arbitrator, cluster is not partition-safe` warnings.

  This fix moves the setting of the timer for arbitration timeout to the `INIT` state, so that the value later read during `FIND` is always initialized. (Bug #17738720)

- The global checkpoint lag watchdog tracking the number of times a check for GCP lag was performed using the system scheduler and used this count to check for a timeout condition, but this caused a number of issues. To overcome these limitations, the GCP watchdog has been refactored to keep track of its own start times, and to calculate elapsed time by reading the (real) clock every time it is called.

  In addition, any backticks (rare in any case) are now handled by taking the backward time as the new current time and calculating the elapsed time for this round as 0. Finally, any ill effects of a forward leap, which possibly could expire the watchdog timer immediately, are reduced by never calculating an elapsed time longer than the requested delay time for the watchdog timer. (Bug #17647469)

  References: See also: Bug #17842035.

- The length of the interval (intended to be 10 seconds) between warnings for `GCP_COMMIT` when the GCP progress watchdog did not detect progress in a global checkpoint was not always calculated correctly. (Bug #17647213)

- In certain rare cases on commit of a transaction, an `Ndb` object was released before the transaction coordinator (`DBTC` kernel block) sent the expected `COMMIT_CONF` signal; `NDB` failed to send a `COMMIT_ACK` signal in response, which caused a memory leak in the `NDB` kernel could later lead to node failure.

  Now an `Ndb` object is not released until the `COMMIT_CONF` signal has actually been received. (Bug #16944817)

- 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 #16890703)

References: See also: Bug #21184102.

- Losing its connections to the management node or data nodes while a query against the `ndbinfo.memoryusage` table was in progress caused the SQL node where the query was issued to fail. (Bug #14483440, Bug #16810415)
- The `ndbd_redo_log_reader` utility now supports a `--help` option. Using this options causes the program to print basic usage information, and then to exit. (Bug #11749591, Bug #36805)

Changes in MySQL NDB Cluster 7.2.14 (5.5.34-ndb-7.2.14) (2013-10-31, General Availability)

MySQL NDB Cluster 7.2.14 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.34 (see Changes in MySQL 5.5.34 (2013-09-20, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Added the `ExtraSendBufferMemory` parameter for management nodes and API nodes. (Formerly, this parameter was available only for configuring data nodes.) See `ExtraSendBufferMemory` (management nodes), and `ExtraSendBufferMemory` (API nodes), for more information. (Bug #14555359)
- `BLOB` and `TEXT` columns are now reorganized by the `ALTER ONLINE TABLE ... REORGANIZE PARTITION` statement. (Bug #13714148)

Bugs Fixed

- **Performance:** In a number of cases found in various locations in the MySQL NDB Cluster codebase, unnecessary iterations were performed; this was caused by failing to break out of a repeating control structure after a test condition had been met. This community-contributed fix removes the unneeded repetitions by supplying the missing breaks. (Bug #16904243, Bug #69392, Bug #16904338, Bug #69394, Bug #16778417, Bug #69171, Bug #16778494, Bug #69172, Bug #16798410, Bug #69207, Bug #16801489, Bug #69215, Bug #16904266, Bug #69393)
- **Packaging; Microsoft Windows:** The MySQL NDB Cluster installer for Windows provided a nonfunctional option to install debug symbols (contained in `*.pdb` files). This option has been removed from the installer.

Note

You can obtain the `*.pdb` debug files for a given MySQL NDB Cluster release from the Windows `.zip` archive for the same release, such as `mysql-`

> `cluster-gpl-7.2.14-win32.zip` or `mysql-cluster-gpl-7.3.2-winx64.zip`.

(Bug #16748308, Bug #69112)

- **Packaging; Microsoft Windows:** The MySQL NDB Cluster Windows installer attempted to use the wrong path to the `my.ini` file and the executables directory. (Bug #13813120, Bug #64510)
- **Packaging:** A fix has been made in this release correcting a number of issues found in some recent `MySQL-Cluster-server` RPM packages, including dependencies on the `mysql-server` package and conflicts with the `mysql-libs` package needed by other common applications. Platforms known to have been affected by these issues include CentOS 6, Red Hat Enterprise Linux 6, and Oracle Linux 6. (Bug #14063590, Bug #14181419, Bug #65534)
- **Microsoft Windows:** The Windows error *`ERROR_FILE_EXISTS`* was not recognized by `NDB`, which treated it as an unknown error. (Bug #16970960)
- **NDB Disk Data:** The statements `CREATE TABLESPACE`, `ALTER LOGFILE GROUP`, and `ALTER TABLESPACE` failed with a syntax error when `INITIAL_SIZE` was specified using letter abbreviations such as `M` or `G`. In addition, `CREATE LOGFILE GROUP` failed when `INITIAL_SIZE`, `UNDO_BUFFER_SIZE`, or both options were specified using letter abbreviations. (Bug #13116514, Bug #16104705, Bug #62858)
- **NDB Replication:** Trying to use a stale `.frm` file and encountering a mismatch bewteen table definitions could cause `mysqld` to make errors when writing to the binary log. (Bug #17250994)
- **NDB Replication:** Replaying a binary log that had been written by a `mysqld` from a MySQL Server distribution (and from not a MySQL NDB Cluster distribution), and that contained DML statements, on a MySQL NDB Cluster SQL node could lead to failure of the SQL node. (Bug #16742250)
- **NDB Cluster APIs:** For each log event retrieved using the MGM API, the log event category (`ndb_mgm_event_category`) was simply cast to an `enum` type, which resulted in invalid category values. Now an offset is added to the category following the cast to ensure that the value does not fall out of the allowed range.

  

  Note

  This change was reverted by the fix for Bug #18354165. See the MySQL NDB Cluster API Developer documentation for `ndb_logevent_get_next()`, for more information.

  (Bug #16723708)

  References: See also: Bug #18354165.

- **NDB Cluster APIs:** The `Event::setTable()` method now supports a pointer or a reference to table as its required argument. If a null table pointer is used, the method now returns -1 to make it clear that this is what has occurred. (Bug #16329082)
- **ndbmemcache:** The `CMakeLists.txt` files for `ndbmemcache` wrote into the source tree, preventing out-of-source builds. (Bug #14650456)
- **ndbmemcache:** `ndbmemcache` did not handle passed-in `BINARY` values correctly.
- Trying to restore to a table having a `BLOB` column in a different position from that of the original one caused `ndb_restore --restore-data` to fail. (Bug #17395298)

- `ndb_restore` could abort during the last stages of a restore using attribute promotion or demotion into an existing table. This could happen if a converted attribute was nullable and the backup had been run on active database. (Bug #17275798)

- The `DBUTIL` data node block is now less strict about the order in which it receives certain messages from other nodes. (Bug #17052422)

- `RealTimeScheduler` did not work correctly with data nodes running `ndbmtd`. (Bug #16961971)

- File system errors occurring during a local checkpoint could sometimes cause an LCP to hang with no obvious cause when they were not handled correctly. Now in such cases, such errors always cause the node to fail. Note that the LQH block always shuts down the node when a local checkpoint fails; the change here is to make likely node failure occur more quickly and to make the original file system error more visible. (Bug #16961443)

- `mysql_upgrade` failed when upgrading from MySQL NDB Cluster 7.1.26 to MySQL NDB Cluster 7.2.13 when it attempted to invoke a stored procedure before the `mysql.proc` table had been upgraded. (Bug #16933405)

  References: This issue is a regression of: Bug #16226274.

- The planned or unplanned shutdown of one or more data nodes while reading table data from the `ndbinfo` database caused a memory leak. (Bug #16932989)

- Maintenance and checking of parent batch completion in the `SPJ` block of the `NDB` kernel was reimplemented. Among other improvements, the completion state of all ancestor nodes in the tree are now preserved. (Bug #16925513)

- Executing `DROP TABLE` while `DBDIH` was updating table checkpoint information subsequent to a node failure could lead to a data node failure. (Bug #16904469)

- In certain cases, when starting a new SQL node, `mysqld` failed with Error 1427 `Api node died, when SUB_START_REQ reached node`. (Bug #16840741)

- Failure to use container classes specific `NDB` during node failure handling could cause leakage of commit-ack markers, which could later lead to resource shortages or additional node crashes. (Bug #16834416)

- Use of an uninitialized variable employed in connection with error handling in the `DBLQH` kernel block could sometimes lead to a data node crash or other stability issues for no apparent reason. (Bug #16834333)

- A race condition in the time between the reception of a `execNODE_FAILREP` signal by the `QMGR` kernel block and its reception by the `DBLQH` and `DBTC` kernel blocks could lead to data node crashes during shutdown. (Bug #16834242)

- The `CLUSTERLOG` command (see Commands in the NDB Cluster Management Client) caused `ndb_mgm` to crash on Solaris SPARC systems. (Bug #16834030)

- The NDB Error-Reporting Utility (`ndb_error_reporter`) failed to include the cluster nodes' log files in the archive it produced when the `FILE` option was set for the parameter `LogDestination`. (Bug #16765651)

  References: See also: Bug #11752792, Bug #44082.

- The LCP fragment scan watchdog periodically checks for lack of progress in a fragment scan performed as part of a local checkpoint, and shuts down the node if there is no progress after a given amount of

time has elapsed. This interval, formerly hard-coded as 60 seconds, can now be configured using the `LcpScanProgressTimeout` data node configuration parameter added in this release.

This configuration parameter sets the maximum time the local checkpoint can be stalled before the LCP fragment scan watchdog shuts down the node. The default is 60 seconds, which provides backward compatibility with previous releases.

You can disable the LCP fragment scan watchdog by setting this parameter to 0. (Bug #16630410)

- Added the `ndb_error_reporter` options `--connection-timeout`, which makes it possible to set a timeout for connecting to nodes, `--dry-scp`, which disables scp connections to remote hosts, and `--skip-nodegroup`, which skips all nodes in a given node group. (Bug #16602002)

  References: See also: Bug #11752792, Bug #44082.

- After issuing `START BACKUP id WAIT STARTED`, if *`id`* had already been used for a backup ID, an error caused by the duplicate ID occurred as expected, but following this, the `START BACKUP` command never completed. (Bug #16593604, Bug #68854)

- `ndb_mgm` treated backup IDs provided to `ABORT BACKUP` commands as signed values, so that backup IDs greater than 2^{31} wrapped around to negative values. This issue also affected out-of-range backup IDs, which wrapped around to negative values instead of causing errors as expected in such cases. The backup ID is now treated as an unsigned value, and `ndb_mgm` now performs proper range checking for backup ID values greater than `MAX_BACKUPS` (2^{32}). (Bug #16585497, Bug #68798)

- When trying to specify a backup ID greater than the maximum allowed, the value was silently truncated. (Bug #16585455, Bug #68796)

- The unexpected shutdown of another data node as a starting data node received its node ID caused the latter to hang in Start Phase 1. (Bug #16007980)

  References: See also: Bug #18993037.

- `SELECT ... WHERE ... LIKE` from an `NDB` table could return incorrect results when using `engine_condition_pushdown=ON`. (Bug #15923467, Bug #67724)

- The `NDB` receive thread waited unnecessarily for additional job buffers to become available when receiving data. This caused the receive mutex to be held during this wait, which could result in a busy wait when the receive thread was running with real-time priority.

  This fix also handles the case where a negative return value from the initial check of the job buffer by the receive thread prevented further execution of data reception, which could possibly lead to communication blockage or configured `ReceiveBufferMemory` underutilization. (Bug #15907515)

- When the available job buffers for a given thread fell below the critical threshold, the internal multithreading job scheduler waited for job buffers for incoming rather than outgoing signals to become available, which meant that the scheduler waited the maximum timeout (1 millisecond) before resuming execution. (Bug #15907122)

- Under some circumstances, a race occurred where the wrong watchdog state could be reported. A new state name `Packing Send Buffers` is added for watchdog state number 11, previously reported as `Unknown place`. As part of this fix, the state numbers for states without names are always now reported in such cases. (Bug #14824490)

- Creating more than 32 hash maps caused data nodes to fail. Usually new hashmaps are created only when performing reorganzation after data nodes have been added or when explicit partitioning is used, such as when creating a table with the `MAX_ROWS` option, or using `PARTITION BY KEY() PARTITIONS n`. (Bug #14710311)

- When a node fails, the Distribution Handler (DBDIH kernel block) takes steps together with the Transaction Coordinator (DBTC) to make sure that all ongoing transactions involving the failed node are taken over by a surviving node and either committed or aborted. Transactions taken over which are then committed belong in the epoch that is current at the time the node failure occurs, so the surviving nodes must keep this epoch available until the transaction takeover is complete. This is needed to maintain ordering between epochs.

  A problem was encountered in the mechanism intended to keep the current epoch open which led to a race condition between this mechanism and that normally used to declare the end of an epoch. This could cause the current epoch to be closed prematurely, leading to failure of one or more surviving data nodes. (Bug #14623333, Bug #16990394)

- When performing an INSERT ... ON DUPLICATE KEY UPDATE on an NDB table where the row to be inserted already existed and was locked by another transaction, the error message returned from the INSERT following the timeout was Transaction already aborted instead of the expected Lock wait timeout exceeded. (Bug #14065831, Bug #65130)

- When using dynamic listening ports for accepting connections from API nodes, the port numbers were reported to the management server serially. This required a round trip for each API node, causing the time required for data nodes to connect to the management server to grow linearly with the number of API nodes. To correct this problem, each data node now reports all dynamic ports at once. (Bug #12593774)

- ndb_error-reporter did not support the --help option. (Bug #11756666, Bug #48606)

  References: See also: Bug #11752792, Bug #44082.

- When START BACKUP WAIT STARTED was run from the command line using ndb_mgm --execute (-e), the client did not exit until the backup completed. (Bug #11752837, Bug #44146)

- Formerly, the node used as the coordinator or leader for distributed decision making between nodes (also known as the DICT manager—see The DBDICT Block) was indicated in the output of the ndb_mgm client SHOW command as the “master” node, although this node has no relationship to a master server in MySQL Replication. (It should also be noted that it is not necessary to know which node is the leader except when debugging NDBCLUSTER source code.) To avoid possible confusion, this label has been removed, and the leader node is now indicated in SHOW command output using an asterisk (*) character. (Bug #11746263, Bug #24880)

- Program execution failed to break out of a loop after meeting a desired condition in a number of internal methods, performing unneeded work in all cases where this occurred. (Bug #69610, Bug #69611, Bug #69736, Bug #17030606, Bug #17030614, Bug #17160263)

- ABORT BACKUP in the ndb_mgm client (see Commands in the NDB Cluster Management Client) took an excessive amount of time to return (approximately as long as the backup would have required to complete, had it not been aborted), and failed to remove the files that had been generated by the aborted backup. (Bug #68853, Bug #17719439)

- Attribute promotion and demotion when restoring data to NDB tables using ndb_restore --restore-data with the --promote-attributes and --lossy-conversions options has been improved as follows:

  - Columns of types CHAR, and VARCHAR can now be promoted to BINARY and VARBINARY, and columns of the latter two types can be demoted to one of the first two.

    Note that converted character data is not checked to conform to any character set.

  - Any of the types CHAR, VARCHAR, BINARY, and VARBINARY can now be promoted to TEXT or BLOB.

When performing such promotions, the only other sort of type conversion that can be performed at the same time is between character types and binary types.

Changes in MySQL NDB Cluster 7.2.13 (5.5.31-ndb-7.2.13) (2013-06-14, General Availability)

MySQL NDB Cluster 7.2.13 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.31 (see Changes in MySQL 5.5.31 (2013-04-18, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **NDB Cluster APIs:** Added `DUMP` code 2514, which provides information about counts of transaction objects per API node. For more information, see DUMP 2514. See also Commands in the NDB Cluster Management Client. (Bug #15878085)
- When `ndb_restore` fails to find a table, it now includes in the error output an NDB API error code giving the reason for the failure. (Bug #16329067)

Bugs Fixed

- **Incompatible Change; NDB Replication:** The default value for the `binlog_format` system variable when the `NDB` storage engine is enabled is now `MIXED`. If `NDB` is not enabled, the default binary logging format is `STATEMENT`, as in the standard MySQL Server. (Bug #16417224)
- **NDB Disk Data:** `NDB` error 899 `RowId already allocated` was raised due to a RowId “leak” which occurred under either of the following sets of circumstances:

  1. Insertion of a row into an in-memory table was rejected after an ordered index update failed due to insufficient `DataMemory`.
  2. Insertion of a row into a Disk Data table was rejected due to lack of sufficient table space.

  (Bug #13927679)

  References: See also: Bug #22494024, Bug #13990924.

- The NDB Error-Reporting Utility (`ndb_error_reporter`) failed to include the cluster nodes' log files in the archive it produced when the `FILE` option was set for the parameter `LogDestination`. (Bug #16765651)

  References: See also: Bug #11752792, Bug #44082.

- A `WHERE` condition that contained a boolean test of the result of an `IN` subselect was not evaluated correctly. (Bug #16678033)

- In some cases a data node could stop with an exit code but no error message other than `(null)` was logged. (This could occur when using `ndbd` or `ndbmtd` for the data node process.) Now in such cases the appropriate error message is used instead (see Data Node Error Messages). (Bug #16614114)
- `mysql_upgrade` failed when run on an SQL node where distributed privileges were in use. (Bug #16226274)
- Improved handling of lagging row change event subscribers by setting size of the GCP pool to the value of `MaxBufferedEpochs`. This fix also introduces a new `MaxBufferedEpochBytes` data node configuration parameter, which makes it possible to set a total number of bytes per node to be reserved for buffering epochs. In addition, a new `DUMP` code (8013) has been added which causes a list a lagging subscribers for each node to be printed to the cluster log (see DUMP 8013). (Bug #16203623)
- Purging the binary logs could sometimes cause `mysqld` to crash. (Bug #15854719)
- An error message in `src/mgmsrv/MgmtSrvr.cpp` was corrected. (Bug #14548052, Bug #66518)
- The help text for `ndb_select_count` did not include any information about using table names. (Bug #11755737, Bug #47551)

Changes in MySQL NDB Cluster 7.2.12 (5.5.30-ndb-7.2.12) (2013-03-27, General Availability)

MySQL NDB Cluster 7.2.12 is a new release of NDB Cluster, incorporating fixes for bugs in previous MySQL NDB Cluster 7.2 releases made in MySQL NDB Cluster 7.2.11 (see Changes in MySQL NDB Cluster 7.2.11 (5.5.29-ndb-7.2.11) (Not released)) and updating the version of the `mysqld` included with the NDB Cluster distribution from version 5.5.29 to 5.5.30 (see Changes in MySQL 5.5.30 (2013-02-05, General Availability)).

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases.

Changes in MySQL NDB Cluster 7.2.11 (5.5.29-ndb-7.2.11) (Not released)

MySQL NDB Cluster 7.2.11 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.29 (see Changes in MySQL 5.5.29 (2012-12-21, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Following an upgrade to MySQL NDB Cluster 7.2.7 or later, it was not possible to downgrade online again to any previous version, due to a change in that version in the default size (number of LDM

threads used) for NDB table hash maps. The fix for this issue makes the size configurable, with the addition of the DefaultHashMapSize configuration parameter.

To retain compatibility with an older release that does not support large hash maps, you can set this parameter in the cluster' config.ini file to the value used in older releases (240) before performing an upgrade, so that the data nodes continue to use smaller hash maps that are compatible with the older release. You can also now employ this parameter in MySQL NDB Cluster 7.0 and MySQL NDB Cluster 7.1 to enable larger hash maps prior to upgrading to MySQL NDB Cluster 7.2. For more information, see the description of the DefaultHashMapSize parameter. (Bug #14800539)

References: See also: Bug #14645319.

Bugs Fixed

- **Important Change; NDB Cluster APIs:** When checking—as part of evaluating an if predicate—which error codes should be propagated to the application, any error code less than 6000 caused the current row to be skipped, even those codes that should have caused the query to be aborted. In addition, a scan that aborted due to an error from DBTUP when no rows had been sent to the API caused DBLQH to send a SCAN_FRAGCONF signal rather than a SCAN_FRAGREF signal to DBTC. This caused DBTC to time out waiting for a SCAN_FRAGREF signal that was never sent, and the scan was never closed.

  As part of this fix, the default ErrorCode value used by NdbInterpretedCode::interpret_exit_nok() has been changed from 899 (Rowid already allocated) to 626 (Tuple did not exist). The old value continues to be supported for backward compatibility. User-defined values in the range 6000-6999 (inclusive) are also now supported. You should also keep in mind that the result of using any other ErrorCode value not mentioned here is not defined or guaranteed.

  See also The NDB Communication Protocol, and NDB Kernel Blocks, for more information. (Bug #16176006)

- **NDB Cluster APIs:** The Ndb::computeHash() API method performs a malloc() if no buffer is provided for it to use. However, it was assumed that the memory thus returned would always be suitably aligned, which is not always the case. Now when malloc() provides a buffer to this method, the buffer is aligned after it is allocated, and before it is used. (Bug #16484617)

- When using tables having more than 64 fragments in a MySQL NDB Cluster where multiple TC threads were configured (on data nodes running ndbmtd, using ThreadConfig), AttrInfo and KeyInfo memory could be freed prematurely, before scans relying on these objects could be completed, leading to a crash of the data node. (Bug #16402744)

  References: See also: Bug #13799800. This issue is a regression of: Bug #14143553.

- When started with --initial and an invalid --config-file (-f) option, ndb_mgmd removed the old configuration cache before verifying the configuration file. Now in such cases, ndb_mgmd first checks for the file, and continues with removing the configuration cache only if the configuration file is found and is valid. (Bug #16299289)

- Executing a DUMP 2304 command during a data node restart could cause the data node to crash with a Pointer too large error. (Bug #16284258)

- Including a table as a part of a pushed join should be rejected if there are outer joined tables in between the table to be included and the tables with which it is joined with; however the check as performed for any such outer joined tables did so by checking the join type against the root of the pushed query, rather than the common ancestor of the tables being joined. (Bug #16199028)

References: See also: Bug #16198866.

- Some queries were handled differently with `ndb_join_pushdown` enabled, due to the fact that outer join conditions were not always pruned correctly from joins before they were pushed down. (Bug #16198866)

  References: See also: Bug #16199028.

- Data nodes could fail during a system restart when the host ran short of memory, due to signals of the wrong types (`ROUTE_ORD` and `TRANSID_AI_R`) being sent to the `DBSPJ` kernel block. (Bug #16187976)

- Attempting to perform additional operations such as `ADD COLUMN` as part of an `ALTER [ONLINE | OFFLINE] TABLE ... RENAME ...` statement is not supported, and now fails with an **`ER_NOT_SUPPORTED_YET`** error. (Bug #16021021)

- The `mysql.server` script exited with an error if the `status` command was executed with multiple servers running. (Bug #15852074)

- Due to a known issue in the MySQL Server, it is possible to drop the `PERFORMANCE_SCHEMA` database. (Bug #15831748) In addition, when executed on a MySQL Server acting as a MySQL NDB Cluster SQL node, `DROP DATABASE` caused this database to be dropped on all SQL nodes in the cluster. Now, when executing a distributed drop of a database, `NDB` does not delete tables that are local only. This prevents MySQL system databases from being dropped in such cases. (Bug #14798043)

  References: See also: Bug #15831748.

- When performing large numbers of DDL statements (100 or more) in succession, adding an index to a table sometimes caused `mysqld` to crash when it could not find the table in `NDB`. Now when this problem occurs, the DDL statement should fail with an appropriate error.

  A workaround in such cases may be to create the table with the index as part of the initial `CREATE TABLE`, rather than adding the index in a subsequent `ALTER TABLE` statement. (Bug #14773491)

- Executing `OPTIMIZE TABLE` on an `NDB` table containing `TEXT` or `BLOB` columns could sometimes cause `mysqld` to fail. (Bug #14725833)

- Executing a `DUMP 1000` command that contained extra or malformed arguments could lead to data node failures. (Bug #14537622)

- Exhaustion of `LongMessageBuffer` memory under heavy load could cause data nodes running `ndbmtd` to fail. (Bug #14488185)

- The `ndb_mgm` client `HELP` command did not show the complete syntax for the `REPORT` command.

Changes in MySQL NDB Cluster 7.2.10 (5.5.29-ndb-7.2.10) (2013-01-02, General Availability)

MySQL NDB Cluster 7.2.10 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.29 (see Changes in MySQL 5.5.29 (2012-12-21, General Availability)).

Functionality Added or Changed

- Added several new columns to the `transporters` table and counters for the `counters` table of the `ndbinfo` information database. The information provided may help in troublehsooting of transport overloads and problems with send buffer memory allocation. For more information, see the descriptions of these tables. (Bug #15935206)
- To provide information which can help in assessing the current state of arbitration in a MySQL NDB Cluster as well as in diagnosing and correcting arbitration problems, 3 new tables—`membership`, `arbitrator_validity_detail`, and `arbitrator_validity_summary`—have been added to the `ndbinfo` information database. (Bug #13336549)

Bugs Fixed

- **NDB Replication:** Setting `slave_allow_batching` had no effect. (Bug #15953730)
- When an `NDB` table grew to contain approximately one million rows or more per partition, it became possible to insert rows having duplicate primary or unique keys into it. In addition, primary key lookups began to fail, even when matching rows could be found in the table by other means.

  This issue was introduced in MySQL NDB Cluster 7.0.36, MySQL NDB Cluster 7.1.26, and MySQL NDB Cluster 7.2.9. Signs that you may have been affected include the following:

  - Rows left over that should have been deleted
  - Rows unchanged that should have been updated
  - Rows with duplicate unique keys due to inserts or updates (which should have been rejected) that failed to find an existing row and thus (wrongly) inserted a new one

  This issue does not affect simple scans, so you can see all rows in a given *`table`* using `SELECT * FROM` *`table`* and similar queries that do not depend on a primary or unique key.

  Upgrading to or downgrading from an affected release can be troublesome if there are rows with duplicate primary or unique keys in the table; such rows should be merged, but the best means of doing so is application dependent.

  In addition, since the key operations themselves are faulty, a merge can be difficult to achieve without taking the MySQL NDB Cluster offline, and it may be necessary to dump, purge, process, and reload the data. Depending on the circumstances, you may want or need to process the dump with an external application, or merely to reload the dump while ignoring duplicates if the result is acceptable.

  Another possibility is to copy the data into another table without the original table' unique key constraints or primary key (recall that `CREATE TABLE t2 SELECT * FROM t1` does not by default copy `t1`'s primary or unique key definitions to `t2`). Following this, you can remove the duplicates from the copy, then add back the unique constraints and primary key definitions. Once the copy is in the desired state, you can either drop the original table and rename the copy, or make a new dump (which can be loaded later) from the copy. (Bug #16023068, Bug #67928)
- The management client command `ALL REPORT BackupStatus` failed with an error when used with data nodes having multiple LQH worker threads (`ndbmtd` data nodes). The issue did not effect the *`node_id`* `REPORT BackupStatus` form of this command. (Bug #15908907)

- The multithreaded job scheduler could be suspended prematurely when there were insufficient free job buffers to allow the threads to continue. The general rule in the job thread is that any queued messages should be sent before the thread is allowed to suspend itself, which guarantees that no other threads or API clients are kept waiting for operations which have already completed. However, the number of messages in the queue was specified incorrectly, leading to increased latency in delivering signals, sluggish response, or otherwise suboptimal performance. (Bug #15908684)

- The setting for the `DefaultOperationRedoProblemAction` API node configuration parameter was ignored, and the default value used instead. (Bug #15855588)

- Node failure during the dropping of a table could lead to the node hanging when attempting to restart.

  When this happened, the `NDB` internal dictionary (`DBDICT`) lock taken by the drop table operation was held indefinitely, and the logical global schema lock taken by the SQL the drop table operation from which the drop operation originated was held until the `NDB` internal operation timed out. To aid in debugging such occurrences, a new dump code, `DUMP 1228` (or `DUMP DictDumpLockQueue`), which dumps the contents of the `DICT` lock queue, has been added in the `ndb_mgm` client. (Bug #14787522)

- Job buffers act as the internal queues for work requests (signals) between block threads in `ndbmtd` and could be exhausted if too many signals are sent to a block thread.

  Performing pushed joins in the `DBSPJ` kernel block can execute multiple branches of the query tree in parallel, which means that the number of signals being sent can increase as more branches are executed. If `DBSPJ` execution cannot be completed before the job buffers are filled, the data node can fail.

  This problem could be identified by multiple instances of the message `sleeploop 10!!` in the cluster out log, possibly followed by `job buffer full`. If the job buffers overflowed more gradually, there could also be failures due to error 1205 (`Lock wait timeout exceeded`), shutdowns initiated by the watchdog timer, or other timeout related errors. These were due to the slowdown caused by the 'sleeploop'.

  Normally up to a 1:4 fanout ratio between consumed and produced signals is permitted. However, since there can be a potentially unlimited number of rows returned from the scan (and multiple scans of this type executing in parallel), any ratio greater 1:1 in such cases makes it possible to overflow the job buffers.

  The fix for this issue defers any lookup child which otherwise would have been executed in parallel with another is deferred, to resume when its parallel child completes one of its own requests. This restricts the fanout ratio for bushy scan-lookup joins to 1:1. (Bug #14709490)

  References: See also: Bug #14648712.

- During an online upgrade, certain SQL statements could cause the server to hang, resulting in the error `Got error 4012 'Request ndbd time-out, maybe due to high load or communication problems' from NDBCLUSTER`. (Bug #14702377)

- The recently added LCP fragment scan watchdog occasionally reported problems with LCP fragment scans having very high table id, fragment id, and row count values.

  This was due to the watchdog not accounting for the time spent draining the backup buffer used to buffer rows before writing to the fragment checkpoint file.

  Now, in the final stage of an LCP fragment scan, the watchdog switches from monitoring rows scanned to monitoring the buffer size in bytes. The buffer size should decrease as data is written to the file, after which the file should be promptly closed. (Bug #14680057)

- Under certain rare circumstances, MySQL NDB Cluster data nodes could crash in conjunction with a configuration change on the data nodes from a single-threaded to a multithreaded transaction coordinator (using the `ThreadConfig` configuration parameter for `ndbmtd`). The problem occurred when a `mysqld` that had been started prior to the change was shut down following the rolling restart of the data nodes required to effect the configuration change. (Bug #14609774)

- On Microsoft Windows with CMake 2.6, the build process would not stop if the `create_initial_db` step failed. (Bug #13713525)

Changes in MySQL NDB Cluster 7.2.9 (5.5.28-ndb-7.2.9) (2012-11-22, General Availability)

Note

MySQL NDB Cluster 7.2.9 was withdrawn shortly after release, due to a problem with primary keys and tables with very many rows that was introduced in this release (Bug #16023068, Bug #67928). Users should upgrade to MySQL NDB Cluster 7.2.10, which fixes this issue.

MySQL NDB Cluster 7.2.9 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.28 (see Changes in MySQL 5.5.28 (2012-09-28, General Availability)).

- Functionality Added or Changed

- Bugs Fixed

Functionality Added or Changed

- **Important Change; MySQL NDB ClusterJ:** A new `CMake` option `WITH_NDB_JAVA` is introduced. When this option is enabled, the MySQL NDB Cluster build is configured to include Java support, including support for `ClusterJ`. If the JDK cannot be found, `CMake` fails with an error. This option is enabled by default; if you do *not* wish to compile Java support into MySQL NDB Cluster, you must now set this explicitly when configuring the build, using `-DWITH_NDB_JAVA=OFF`. (Bug #12379755)

- Added 3 new columns to the `transporters` table in the `ndbinfo` database. The `remote_address`, `bytes_sent`, and `bytes_received` columns help to provide an overview of data transfer across the transporter links in a MySQL NDB Cluster. This information can be useful in verifying system balance, partitioning, and front-end server load balancing; it may also be of help when diagnosing network problems arising from link saturation, hardware faults, or other causes. (Bug #14685458)

- Data node logs now provide tracking information about arbitrations, including which nodes have assumed the arbitrator role and at what times. (Bug #11761263, Bug #53736)

Bugs Fixed

- **NDB Disk Data:** Concurrent DML and DDL operations against the same NDB table could cause `mysqld` to crash. (Bug #14577463)

- **NDB Replication:** When the value of `ndb_log_apply_status` was set to 1, it was theoretically possible for the `ndb_apply_status` table's `server_id` column not to be propagated correctly. (Bug #14772503)

- **NDB Replication:** Transactions originating on a replication master are applied on slaves as if using `AO_AbortError`, but transactions replayed from a binary log were not. Now transactions being replayed from a log are handled in the same way as those coming from a “live” replication master.

  See NdbOperation::AbortOption, for more information. (Bug #14615095)

- A slow filesystem during local checkpointing could exert undue pressure on `DBDIH` kernel block file page buffers, which in turn could lead to a data node crash when these were exhausted. This fix limits the number of table definition updates that `DBDIH` can issue concurrently. (Bug #14828998)

- The management server process, when started with `--config-cache=FALSE`, could sometimes hang during shutdown. (Bug #14730537)

- The output from `ndb_config --configinfo` now contains the same information as that from `ndb_config --configinfo --xml`, including explicit indicators for parameters that do not require restarting a data node with `--initial` to take effect. In addition, `ndb_config` indicated incorrectly that the `LogLevelCheckpoint` data node configuration parameter requires an initial node restart to take effect, when in fact it does not; this error was also present in the MySQL NDB Cluster documentation, where it has also been corrected. (Bug #14671934)

- Attempting to restart more than 5 data nodes simultaneously could cause the cluster to crash. (Bug #14647210)

- In MySQL NDB Cluster 7.2.7, the size of the hash map was increased to 3840 LDM threads. However, when upgrading a MySQL NDB Cluster from a previous release, existing tables could not use or be modified online to take advantage of the new size, even when the number of fragments was increased by (for example) adding new data nodes to the cluster. Now in such cases, following an upgrade and (once the number of fragments has been increased), you can run `ALTER TABLE ... REORGANIZE PARTITION` on tables that were created in MySQL NDB Cluster 7.2.6 or earlier, after which they can use the larger hash map size. (Bug #14645319)

- Concurrent `ALTER TABLE` with other DML statements on the same NDB table returned `Got error -1 'Unknown error code' from NDBCLUSTER`. (Bug #14578595)

- CPU consumption peaked several seconds after the forced termination an NDB client application due to the fact that the DBTC kernel block waited for any open transactions owned by the disconnected API client to be terminated in a busy loop, and did not break between checks for the correct state. (Bug #14550056)

- Receiver threads could wait unnecessarily to process incomplete signals, greatly reducing performance of `ndbmtd`. (Bug #14525521)

- On platforms where epoll was not available, setting multiple receiver threads with the `ThreadConfig` parameter caused `ndbmtd` to fail. (Bug #14524939)

- Setting `BackupMaxWriteSize` to a very large value as compared with `DiskCheckpointSpeed` caused excessive writes to disk and CPU usage. (Bug #14472648)

- Added the `--connect-retries` and `--connect-delay` startup options for `ndbd` and `ndbmtd`. `--connect-retries` (default 12) controls how many times the data node tries to connect to a management server before giving up; setting it to -1 means that the data node never stops trying to make contact. `--connect-delay` sets the number of seconds to wait between retries; the default is 5. (Bug #14329309, Bug #66550)

- Following a failed `ALTER TABLE ... REORGANIZE PARTITION` statement, a subsequent execution of this statement after adding new data nodes caused a failure in the `DBDIH` kernel block which led to an unplanned shutdown of the cluster.

  `DUMP` code 7019 was added as part of this fix. It can be used to obtain diagnostic information relating to a failed data node. See DUMP 7019, for more information. (Bug #14220269)

  References: See also: Bug #18550318.

- It was possible in some cases for two transactions to try to drop tables at the same time. If the master node failed while one of these operations was still pending, this could lead either to additional node failures (and cluster shutdown) or to new dictionary operations being blocked. This issue is addressed by ensuring that the master will reject requests to start or stop a transaction while there are outstanding dictionary takeover requests. In addition, table-drop operations now correctly signal when complete, as the `DBDICT` kernel block could not confirm node takeovers while such operations were still marked as pending completion. (Bug #14190114)

- The `DBSPJ` kernel block had no information about which tables or indexes actually existed, or which had been modified or dropped, since execution of a given query began. Thus, `DBSPJ` might submit dictionary requests for nonexistent tables or versions of tables, which could cause a crash in the `DBDIH` kernel block.

  This fix introduces a simplified dictionary into the `DBSPJ` kernel block such that `DBSPJ` can now check reliably for the existence of a particular table or version of a table on which it is about to request an operation. (Bug #14103195)

- Previously, it was possible to store a maximum of 46137488 rows in a single MySQL NDB Cluster partition. This limitation has now been removed. (Bug #13844405, Bug #14000373)

  References: See also: Bug #13436216.

- When using `ndbmtd` and performing joins, data nodes could fail where `ndbmtd` processes were configured to use a large number of local query handler threads (as set by the `ThreadConfig` configuration parameter), the tables accessed by the join had a large number of partitions, or both. (Bug #13799800, Bug #14143553)

Changes in MySQL NDB Cluster 7.2.8 (5.5.27-ndb-7.2.8) (2012-09-07, General Availability)

MySQL NDB Cluster 7.2.8 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.27 (see Changes in MySQL 5.5.27 (2012-08-02, General Availability)).

Bugs Fixed

- An improvement introduced in MySQL NDB Cluster 7.2.7 saves memory used by buffers for communications between threads. One way in which this was implemented was by not allocating buffers between threads which were assumed not to communicate, including communication between local query handler (LQH or LDM) threads.

However, the BACKUP kernel block is used by the LDM thread, and during NDB native backup the first instance of this block used by the first LDM thread acts as a client coordinator, and thus attempts to communicate with other LDM threads. This caused the START BACKUP command to fail when using ndbmtd configured with multiple LDM threads.

The fix for this issue restores the buffer used for communication between LDM threads in such cases. (Bug #14489398)

- When reloading the redo log during a node or system restart, and with NoOfFragmentLogFiles greater than or equal to 42, it was possible for metadata to be read for the wrong file (or files). Thus, the node or nodes involved could try to reload the wrong set of data. (Bug #14389746)

- When an NDB table was created during a data node restart, the operation was rolled back in the NDB engine, but not on the SQL node where it was executed. This was due to the table .FRM files not being cleaned up following the operation that was rolled back by NDB. Now in such cases these files are removed. (Bug #13824846)

Changes in MySQL NDB Cluster 7.2.7 (5.5.25a-ndb-7.2.7) (2012-07-17, General Availability)

MySQL NDB Cluster 7.2.7 is a new release of NDB Cluster, incorporating new features in the NDB storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.25a (see Changes in MySQL 5.5.25a (2012-07-05, General Availability)).

Bugs Fixed

- **Important Change:** When FILE was used for the value of the LogDestination parameter without also specifying the filename, the log file name defaulted to logger.log. Now in such cases, the name defaults to ndb_*nodeid*_cluster.log. (Bug #11764570, Bug #57417)

- **Packaging; Solaris:** Some builds on Solaris 64-bit failed because the packages exceeded the 2GB limit for the SVR4 installation layout. Now such packages are built without the embedded versions of mysqltest and mysql-client_test to save space. (Bug #14058643)

- **Packaging; NDB Cluster APIs:** The file META-INF/services/org.apache.openjpa.lib.conf.ProductDerivation was missing from the clusterjpa JAR file. This could cause setting openjpa.BrokerFactory to “ndb” to be rejected. (Bug #14192154)

  References: See also: Bug #52106.

- **NDB Cluster APIs:** libndbclient did not include the NdbScanFilter class. (Bug #14010507)

- **NDB Cluster APIs:** When an NDB API application called NdbScanOperation::nextResult() again after the previous call had returned end-of-file (return code 1), a transaction object was leaked. Now when this happens, NDB returns error code 4210 (Ndb sent more info than length specified); previouslyu in such cases, -1 was returned. In addition, the extra transaction object associated with the scan is freed, by returning it to the transaction coordinator's idle list. (Bug #11748194)

- If the Transaction Coordinator aborted a transaction in the “prepared” state, this could cause a resource leak. (Bug #14208924)
- When attempting to connect using a socket with a timeout, it was possible (if the timeout was exceeded) for the socket not to be set back to blocking. (Bug #14107173)
- A shortage of scan fragment records in `DBTC` resulted in a leak of concurrent scan table records and key operation records. (Bug #13966723)
- Attempting to add both a column and an index on that column in the same online `ALTER TABLE` statement caused `mysqld` to fail. Although this issue affected only the `mysqld` shipped with MySQL NDB Cluster, the table named in the `ALTER TABLE` could use any storage engine for which online operations are supported. (Bug #12755722)

Changes in MySQL NDB Cluster 7.2.6 (5.5.22-ndb-7.2.6) (2012-05-21, General Availability)

MySQL NDB Cluster 7.2.6 is a new release of NDB Cluster, incorporating new features in the `NDB` storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.22 (see Changes in MySQL 5.5.22 (2012-03-21, General Availability)).

Bugs Fixed

- **Important Change:** The `ALTER ONLINE TABLE ... REORGANIZE PARTITION` statement can be used to create new table partitions after new empty nodes have been added to a MySQL NDB Cluster. Usually, the number of partitions to create is determined automatically, such that, if no new partitions are required, then none are created. This behavior can be overridden by creating the original table using the `MAX_ROWS` option, which indicates that extra partitions should be created to store a large number of rows. However, in this case `ALTER ONLINE TABLE ... REORGANIZE PARTITION` simply uses the `MAX_ROWS` value specified in the original `CREATE TABLE` statement to determine the number of partitions required; since this value remains constant, so does the number of partitions, and so no new ones are created. This means that the table is not rebalanced, and the new data nodes remain empty.

  To solve this problem, support is added for `ALTER ONLINE TABLE ... MAX_ROWS=`*`newvalue`*, where *`newvalue`* is greater than the value used with `MAX_ROWS` in the original `CREATE TABLE` statement. This larger `MAX_ROWS` value implies that more partitions are required; these are allocated on the new data nodes, which restores the balanced distribution of the table data.

  For more information, see ALTER TABLE Statement, and Adding NDB Cluster Data Nodes Online. (Bug #13714648)

- **NDB Replication:** Error handling in conflict detection and resolution has been improved to include errors generated for reasons other than operation execution errors, and to distinguish better between permanent errors and transient errors. Transactions failing due to transient problems are now retried rather than leading to SQL node shutdown as occurred in some cases. (Bug #13428909)
- **NDB Replication:** DDL statements could sometimes be missed during replication channel cutover, due to the fact that there may not be any epochs following the last applied epoch when the slave is up to date and no new epoch has been finalized on the master. Because epochs are not consecutively numbered, there may be a gap between the last applied epoch and the next epoch; thus it is not possible

to determine the number assigned to the next epoch. This meant that, if the new master did not have all epochs, it was possible for those epochs containing only DDL statements to be skipped over.

The fix for this problem includes modifications to mysqld binary logging code so that the next position in the binary log following the `COMMIT` event at the end of an epoch transaction, as well as the addition of two new columns `next_file` and `next_position` to the `mysql.ndb_binlog_index` table. In addition, a new replication channel cutover mechanism is defined that employs these new columns. To make use of the new cutover mechanism, it is necessary to modify the query used to obtain the start point; in addition, to simplify prevention of possible errors caused by duplication of DDL statements, a new shorthand value `ddl_exist_errors` is implemented for use with the `mysqld` option `--slave-skip-errors`. It is highly recommended that you use this option and value on the new replication slave when using the modified query.

For more information, see Implementing Failover with NDB Cluster Replication.

Note that the existing replication channel cutover mechanism continues to function as before, including the same limitations described previously. (Bug #11762277, Bug #54854)

- **NDB Cluster APIs:** An assert in `memcache/include/Queue.h` by `NDB` could cause `memcached` to fail. (Bug #13874027)
- An error handling routine in the local query handler (`DBLQH`) used the wrong code path, which could corrupt the transaction ID hash, causing the data node process to fail. This could in some cases possibly lead to failures of other data nodes in the same node group when the failed node attempted to restart. (Bug #14083116)
- When a fragment scan occurring as part of a local checkpoint (LCP) stopped progressing, this kept the entire LCP from completing, which could result it redo log exhaustion, write service outage, inability to recover nodes, and longer system recovery times. To help keep this from occurring, MySQL NDB Cluster now implements an LCP watchdog mechanism, which monitors the fragment scans making up the LCP and takes action if the LCP is observed to be delinquent.

  This is intended to guard against any scan related system-level I/O errors or other issues causing problems with LCP and thus having a negative impact on write service and recovery times. Each node independently monitors the progress of local fragment scans occurring as part of an LCP. If no progress is made for 20 seconds, warning logs are generated every 10 seconds thereafter for up to 1 minute. At this point, if no progress has been made, the fragment scan is considered to have hung, and the node is restarted to enable the LCP to continue.

  In addition, a new `ndbd` exit code `NDBD_EXIT_LCP_SCAN_WATCHDOG_FAIL` is added to identify when this occurs. See LQH Errors, for more information. (Bug #14075825)
- It could sometimes happen that a query pushed down to the data nodes could refer to buffered rows which had been released, and possibly overwritten by other rows. Such rows, if overwritten, could lead to incorrect results from a pushed query, and possibly even to failure of one or more data nodes or SQL nodes. (Bug #14010406)
- `DUMP 2303` in the `ndb_mgm` client now includes the status of the single fragment scan record reserved for a local checkpoint. (Bug #13986128)
- Pushed joins performed as part of a stored procedure or trigger could cause spurious `Out of memory` errors on the SQL node where they were executed. (Bug #13945264)

  References: See also: Bug #13944272.
- `INSERT ... SELECT` executed inside a trigger or stored procedure with `ndb_join_pushdown` enabled could lead to a crash of the SQL node on which it was run. (Bug #13901890)

References: See also: Bug #13945264.

- When upgrading or downgrading between a MySQL NDB Cluster version supporting distributed pushdown joins (MySQL NDB Cluster 7.2 and later) and one that did not, queries that the later MySQL NDB Cluster version tried to push down could cause data nodes still running the earlier version to fail. Now the SQL nodes check the version of the software running on the data nodes, so that queries are not pushed down if there are any data nodes in the cluster that do not support pushdown joins. (Bug #13894817)

- `ndbmemcached` exited unexpectedly when more than 128 clients attempted to connect concurrently using prefixes. In addition, a `NOT FOUND` error was returned when the `memcached` engine encountered a temporary error from `NDB`; now the error `No Ndb Instances in freelist` is returned instead. (Bug #13890064, Bug #13891085)

- The performance of `ndbmemcache` with a workload that consisted mostly of primary key reads became degraded. (Bug #13868787, Bug #64713)

- When the `--skip-config-cache` and `--initial` options were used together, `ndb_mgmd` failed to start. (Bug #13857301)

- The memcached server failed to build correctly on 64-bit Solaris/SPARC. (Bug #13854122)

- `ALTER ONLINE TABLE` failed when a `DEFAULT` option was used. (Bug #13830980)

- In some cases, restarting data nodes spent a very long time in Start Phase 101, when API nodes must connect to the starting node (using `NdbEventOperation`), when the API nodes trying to connect failed in a live-lock scenario. This connection process uses a handshake during which a small number of messages are exchanged, with a timeout used to detect failures during the handshake.

  Prior to this fix, this timeout was set such that, if one API node encountered the timeout, all other nodes connecting would do the same. The fix also decreases this timeout. This issue (and the effects of the fix) are most likely to be observed on relatively large configurations having 10 or more data nodes and 200 or more API nodes. (Bug #13825163)

- `ndbmtd` failed to restart when the size of a table definition exceeded 32K.

  (The size of a table definition is dependent upon a number of factors, but in general the 32K limit is encountered when a table has 250 to 300 columns.) (Bug #13824773)

- An initial start using `ndbmtd` could sometimes hang. This was due to a state which occurred when several threads tried to flush a socket buffer to a remote node. In such cases, to minimize flushing of socket buffers, only one thread actually performs the send, on behalf of all threads. However, it was possible in certain cases for there to be data in the socket buffer waiting to be sent with no thread ever being chosen to perform the send. (Bug #13809781)

- When trying to use `ndb_size.pl --hostname=`*`host`*`:`*`port`* to connect to a MySQL server running on a nonstandard port, the *`port`* argument was ignored. (Bug #13364905, Bug #62635)

- The `transaction_allow_batching` server system variable was inadvertently removed from the NDB 7.2 codebase prior to General Availability. This fix restores the variable. (Bug #64697, Bug #13891116, Bug #13947227)

Changes in MySQL NDB Cluster 7.2.5 (5.5.20-ndb-7.2.5) (2012-03-20, General Availability)

MySQL NDB Cluster 7.2.5 is a new release of NDB Cluster, incorporating new features in the NDB storage engine, and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.20 (see Changes in MySQL 5.5.20 (2012-01-10, General Availability)).

Bugs Fixed

- **Important Change; NDB Replication:** The master limited the number of operations per transaction to 10000 (based on TimeBetweenEpochs). This could result in a larger number of data-modification operations in a single epoch than could be applied at one time, due to the limit imposed on the slave by its (own) setting for MaxDMLOperationsPerTransaction.

  The fix for this issue is to allow a replication slave cluster to exceed the configured value of MaxDMLOperationsPerTransaction when necessary, so that it can apply all DML operations received from the master in the same transaction. (Bug #12825405)

- **Important Change:** A number of changes have been made in the configuration of transporter send buffers.

  1. The data node configuration parameter ReservedSendBufferMemory is now deprecated, and thus subject to removal in a future MySQL NDB Cluster release. ReservedSendBufferMemory has been non-functional since it was introduced and remains so.

  2. TotalSendBufferMemory now works correctly with data nodes using ndbmtd.

  3. SendBufferMemory can now over-allocate into SharedGlobalMemory for ndbmtd data nodes (only).

  4. A new data node configuration parameter ExtraSendBufferMemory is introduced. Its purpose is to control how much additional memory can be allocated to the send buffer over and above that specified by TotalSendBufferMemory or SendBufferMemory. The default setting (0) allows up to 16MB to be allocated automatically.

  (Bug #13633845, Bug #11760629, Bug #53053)

- **NDB Replication:** Conflict detection and resolution for statements updating a given table could be employed only on the same server where the table was created. When an NDB table is created by executing DDL on an SQL node, the binary log setup portion of the processing for the CREATE TABLE statement reads the table's conflict detection function from the ndb_replication table and sets up that function for the table. However, when the created table was discovered by other SQL nodes attached to the same MySQL NDB Cluster due to schema distribution, the conflict detection function was not correctly set up. The same problem occurred when an NDB table was discovered as a result of selecting a database for the first time (such as when executing a USE statement), and when a table was discovered as a result of scanning all files at server startup.

  Both of these issues were due to a dependency of the conflict detection and resolution code on table objects, even in cases where checking for such objects might not be appropriate. With this fix, conflict

detection and resolution for any NDB table works whether the table was created on the same SQL node, or on a different one. (Bug #13578660)

- Setting insert_id had no effect on the auto-increment counter for NDB tables. (Bug #13731134)
- A data node crashed when more than 16G fixed-size memory was allocated by DBTUP to one fragment (because the DBACC kernel block was not prepared to accept values greater than 32 bits from it, leading to an overflow). Now in such cases, the data node returns Error 889 Table fragment fixed data reference has reached maximum possible value.... When this happens, you can work around the problem by increasing the number of partitions used by the table (such as by using the MAXROWS option with CREATE TABLE). (Bug #13637411)

  References: See also: Bug #11747870, Bug #34348.
- Several instances in the NDB code affecting the operation of multithreaded data nodes, where SendBufferMemory was associated with a specific thread for an unnecessarily long time, have been identified and fixed, by minimizing the time that any of these buffers can be held exclusively by a given thread (send buffer memory being critical to operation of the entire node). (Bug #13618181)
- A very large value for BackupWriteSize, as compared to BackupMaxWriteSize, BackupDataBufferSize, or BackupLogBufferSize, could cause a local checkpoint or backup to hang. (Bug #13613344)
- Queries using LIKE ... ESCAPE on NDB tables failed when pushed down to the data nodes. Such queries are no longer pushed down, regardless of the value of engine_condition_pushdown. (Bug #13604447, Bug #61064)
- To avoid TCP transporter overload, an overload flag is kept in the NDB kernel for each data node; this flag is used to abort key requests if needed, yielding error 1218 Send Buffers overloaded in NDB kernel in such cases. Scans can also put significant pressure on transporters, especially where scans with a high degree of parallelism are executed in a configuration with relatively small send buffers. However, in these cases, overload flags were not checked, which could lead to node failures due to send buffer exhaustion. Now, overload flags are checked by scans, and in cases where returning sufficient rows to match the batch size (--ndb-batch-size server option) would cause an overload, the number of rows is limited to what can be accommodated by the send buffer.

  See also Configuring NDB Cluster Send Buffer Parameters. (Bug #13602508)
- A SELECT from an NDB table using LIKE with a multibyte column (such as utf8) did not return the correct result when engine_condition_pushdown was enabled. (Bug #13579318, Bug #64039)

  References: See also: Bug #13608135.
- memcached started with the ndb_engine.so plugin failed to start if the ndbmemcache database was missing. (Bug #13567414)
- When starting memcached using ndb_engine.so with only 2 API slots available in the cluster configuration file, memcached attempted to make a third connection and crashed when this failed. (Bug #13559728)
- A node failure and recovery while performing a scan on more than 32 partitions led to additional node failures during node takeover. (Bug #13528976)
- The --skip-config-cache option now causes ndb_mgmd to skip checking for the configuration directory, and thus to skip creating it in the event that it does not exist. (Bug #13428853)

Changes in MySQL NDB Cluster 7.2.4 (5.5.19-ndb-7.2.4) (2012-02-15, General Availability)

MySQL NDB Cluster 7.2.4 is the first *General Availability* release in the MySQL NDB Cluster 7.2 release series, incorporating new features in the `NDB` storage engine and fixing recently discovered bugs in previous MySQL NDB Cluster 7.2 development releases.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.19 (see Changes in MySQL 5.5.19 (2011-12-08, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change:** A `mysqld` process joining a MySQL NDB Cluster where distributed privileges are in use now automatically executes a `FLUSH PRIVILEGES` as part of the connection process, so that the cluster's distributed privileges take immediate effect on the new SQL node. (Bug #13340854)

Bugs Fixed

- **Packaging:** RPM distributions of MySQL NDB Cluster 7.1 contained a number of packages which in MySQL NDB Cluster 7.2 have been merged into the `MySQL-Cluster-server` RPM. However, the MySQL NDB Cluster 7.2 `MySQL-Cluster-server` RPM did not actually obsolete these packages, which meant that they had to be removed manually prior to performing an upgrade from a MySQL NDB Cluster 7.1 RPM installation.

  These packages include the `MySQL-Cluster-clusterj`, `MySQL-Cluster-extra`, `MySQL-Cluster-management`, `MySQL-Cluster-storage`, and `MySQL NDB Cluster-tools` RPMs.

  For more information, see Installing NDB Cluster from RPM. (Bug #13545589)

- **NDB Replication:** Under certain circumstances, the `Rows` count in the output of `SHOW TABLE STATUS` for a replicated slave `NDB` table could be misreported as many times larger than the result of `SELECT COUNT(*)` on the same table. (Bug #13440282)

- Accessing a table having a `BLOB` column but no primary key following a restart of the SQL node failed with Error 1 (`Unknown error code`). (Bug #13563280)

- At the beginning of a local checkpoint, each data node marks its local tables with a “to be checkpointed” flag. A failure of the master node during this process could cause either the LCP to hang, or one or more data nodes to be forcibly shut down. (Bug #13436481)

- A node failure while a `ANALYZE TABLE` statement was executing resulted in a hung connection (and the user was not informed of any error that would cause this to happen). (Bug #13416603)

  References: See also: Bug #13407848.

Changes in MySQL NDB Cluster 7.2.3 (5.5.17-ndb-7.2.3) (Not released)

MySQL NDB Cluster 7.2.3 is a new development preview release of NDB Cluster, incorporating new features in the NDB storage engine for testing and user feedback.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.17 (see Changes in MySQL 5.5.17 (2011-10-19, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Added the ThreadConfig data node configuration parameter to enable control of multiple threads and CPUs when using ndbmtd, by assigning threads of one or more specified types to execute on one or more CPUs. This can provide more precise and flexible control over multiple threads than can be obtained using the LockExecuteThreadToCPU parameter. (Bug #11795581)

Bugs Fixed

- **Important Change; NDB Replication:** A unique key constraint violation caused NDB slaves to stop rather than to continue when the slave_exec_mode was IDEMPOTENT. In such cases, NDB now behaves as other MySQL storage engines do when in IDEMPOTENT mode. (Bug #11756310)
- **NDB Replication:** With many SQL nodes, all writing binary logs, connected to a MySQL NDB Cluster, RENAME TABLE could cause data node processes (ndbmtd) to fail. (Bug #13447705)
- Added the MinFreePct data node configuration parameter, which specifies a percentage of data node resources to hold in reserve for restarts. The resources monitored are DataMemory, IndexMemory, and any per-table MAX_ROWS settings (see CREATE TABLE Statement). The default value of MinFreePct is 5, which means that 5% from each these resources is now set aside for restarts. (Bug #13436216)
- Issuing TRUNCATE TABLE on mysql.user, mysql.host, mysql.db, mysql.tables_priv, mysql.proxies_priv, or mysql.procs_priv, when these tables had been converted to MySQL NDB Cluster distributed grant tables, caused mysqld to crash. (Bug #13346955)
- Restarting an SQL node configured for distributed grants could sometimes result in a crash. (Bug #13340819)
- Previously, forcing simultaneously the shutdown of multiple data nodes using SHUTDOWN -F in the ndb_mgm management client could cause the entire cluster to fail. Now in such cases, any such nodes are forced to abort immediately. (Bug #12928429)

Changes in MySQL NDB Cluster 7.2.2 (5.5.16-ndb-7.2.2) (2011-12-14, Development Milestone Release)

MySQL NDB Cluster 7.2.2 is a new development preview release of NDB Cluster, incorporating new features in the NDB storage engine for testing and user feedback.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.16 (see Changes in MySQL 5.5.16 (2011-09-15, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **NDB Cluster APIs:** Added support for the Memcache API using *ndbmemcache*, a loadable storage engine for memcached version 1.6 and later, which can be used to provide a persistent MySQL NDB Cluster data store, accessed using the memcache protocol.

  The standard `memcached` caching engine is now included in the MySQL NDB Cluster distribution. Each `memcached` server, in addition to providing direct access to data stored in a MySQL NDB Cluster, is able to cache data locally and serve (some) requests from this local cache.

  The `memcached` server can also provide an interface to existing MySQL NDB Cluster tables that is strictly defined, so that an administrator can control exactly which tables and columns are referenced by particular memcache keys and values, and which operations are allowed on these keys and values.

  For more information, see ndbmemcache—Memcache API for NDB Cluster.

- Added the `ndbinfo_select_all` utility.

Bugs Fixed

- **Microsoft Windows:** The `include/storage` directory, where the header files supplied for use in compiling MySQL NDB Cluster applications are normally located, was missing from MySQL NDB Cluster release packages for Windows. (Bug #12690665)

- **NDB Replication:** The `mysqlbinlog --database` option generated table mapping errors when used with `NDB` tables, unless the binary log was generated using `--log-bin-use-v1-row-events=0`. (Bug #13067813)

- When adding data nodes online, if the SQL nodes were not restarted before starting the new data nodes, the next query to be executed crashed the SQL node on which it was run. (Bug #13715216, Bug #62847)

  References: This issue is a regression of: Bug #13117187.

Changes in MySQL NDB Cluster 7.2.1 (5.5.15-ndb-7.2.1) (2011-10-03, Development Milestone Release)

MySQL NDB Cluster 7.2.1 is a new development preview release of NDB Cluster, incorporating new features in the `NDB` storage engine for testing and user feedback.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.5 through MySQL 5.5.15 (see Changes in MySQL 5.5.15 (2011-07-28, General Availability)).

Functionality Added or Changed

- **Important Change; NDB Replication:** Due to the existing layout of binary log row events, it was not possible to extend them with extra information which could be safely ignored by old slaves. New versions of the `WRITE_ROW`, `UPDATE_ROW`, and `DELETE_ROW` events have been implemented; these are referred to as “Version 2” binary log events, and are intended for future enhancements such as improved conflict detection and resolution. The `TABLE_MAP` event is not affected.

  Version 2 binary log row events are not backward compatible, and cannot be read by older slaves. A new `mysqld` option `--log-bin-use-v1-row-events` can be used to force writing of Version 1 row events into the binary log. This can be used during upgrades to make a newer `mysqld` generate Version 1 binary log row events that can be read by older slaves.

- **Important Change:** By default, data nodes now exhibit fail-fast behavior whenever they encounter corrupted tuples—in other words, a data node forcibly shuts down whenever it detects a corrupted tuple. To override this behavior, you can disable the `CrashOnCorruptedTuple` data node configuration parameter, which is enabled by default.

  This is a change from MySQL NDB Cluster 7.0 and MySQL NDB Cluster 7.1, where `CrashOnCorruptedTuple` was disabled (so that data nodes ignored tuple corruption) by default. (Bug #12598636)

- **Important Change:** The default values for a number of MySQL NDB Cluster data node configuration parameters have changed, to provide more resiliency to environmental issues and better handling of some potential failure scenarios, and to perform more reliably with increases in memory and other resource requirements brought about by recent improvements in join handling by `NDB`. The affected parameters are listed here:

  - `HeartbeatIntervalDbDb`: Default increased from 1500 ms to 5000 ms.
  - `ArbitrationTimeout`: Default increased from 3000 ms to 7500 ms.
  - `TimeBetweenEpochsTimeout`: Now effectively disabled by default (default changed from 4000 ms to 0).
  - `SharedGlobalMemory`: Default increased from 20MB to 128MB.
  - `MaxParallelScansPerFragment`: Default increased from 32 to 256.

  In addition, when `MaxNoOfLocalScans` is not specified, the value computed for it automatically has been increased by a factor of 4 (that is, to 4 times `MaxNoOfConcurrentScans`, times the number of data nodes in the cluster).

- **Important Change:** MySQL user privileges can now be distributed automatically across all MySQL servers (SQL nodes) connected to the same MySQL NDB Cluster. Previously, each MySQL Server's user privilege tables were required to use the `MyISAM` storage engine, which meant that a user account and its associated privileges created on one SQL node were not available on any other SQL node without manual intervention. MySQL NDB Cluster now provides an SQL script file `ndb_dist_priv.sql` that can be found in `share/mysql` under the MySQL installation directory; loading this script creates a stored procedure `mysql_cluster_move_privileges` that can be used following initial installation to convert the privilege tables to the `NDB` storage engine, so that any time a

MySQL user account is created, dropped, or has its privileges updated on any SQL node, the changes take effect immediately on all other MySQL servers attached to the cluster. Note that you may need to execute `FLUSH PRIVILEGES` on any SQL nodes with connected MySQL clients (or to disconnect and then reconnect the clients) in order for those clients to be able to see the changes in privileges.

Once `mysql_cluster_move_privileges` has been executed successfully, all MySQL user privileges are distributed across all connected MySQL Servers. MySQL Servers that join the cluster after this automatically participate in the privilege distribution.

`ndb_dist_priv.sql` also provides stored routines that can be used to verify that the privilege tables have been distributed successfully, and to perform other tasks.

For more information, see Distributed Privileges Using Shared Grant Tables.

- **Performance:** Added support for pushing down joins to the NDB kernel for parallel execution across the data nodes, which can speed up execution of joins across `NDB` tables by a factor of 20 or more in some cases. Some restrictions apply on the types of joins that can be optimized in this way; in particular, columns to be joined must use exactly the same data type, and cannot be any of the `BLOB` or `TEXT` types. In addition, columns to be joined must be part of a table index or primary key. Support for this feature can be enabled or disabled using the `ndb_join_pushdown` server system variable (enabled by default); see the description of this variable for more information and examples.

  As part of this improvement, the status variables `Ndb_pushed_queries_defined`, `Ndb_pushed_queries_dropped`, `Ndb_pushed_queries_executed`, and `Ndb_pushed_reads` also been introduced for monitoring purposes. You can also see whether a given join is pushed down using `EXPLAIN`. In addition, several new counters relating to push-down join performance have been added to the `counters` table in the `ndbinfo` database. For more information, see the descriptions of the status variables previously mentioned.

- **NDB Replication:** Added two new conflict detection functions `NDB$EPOCH()` and `NDB$EPOCH_TRANS()` useful in circular replication scenarios with two MySQL NDB Clusters. Each of these functions requires designating one cluster as primary and one as secondary, and implements a “primary always wins” rule for determining whether to accept conflicting changes. When using `NDB$EPOCH()`, conflicting rows on the secondary are realigned with those on the primary; when using `NDB$EPOCH_TRANS()`, it is transactions containing rows in conflict (and any transactions which depend on them) on the secondary that are realigned.

  Using `NDB$EPOCH_TRANS()` as the conflict detection function has two additional requirements:

  1. The binary log must be written using Version 2 row events; that is, the `mysqld` processes on both the primary and the secondary must be started with `--log-bin-use-v1-row-events=0`.

  2. The secondary must include transaction IDs for all rows written into its binary log. This can be done by setting `--ndb-log-transaction-id=1`. (This server option is also added in this release.)

  A number of new server status variables have been added to monitor conflict detection and resolution performed using these functions, including `Ndb_conflict_fn_epoch`, `Ndb_conflict_fn_epoch_trans`, and `Ndb_conflict_trans_row_conflict_count`. See NDB Cluster Status Variables.

  For more information, see NDB Cluster Replication Conflict Resolution.

- It is now possible to filter the output from `ndb_config` so that it displays only system, data node, or connection parameters and values, using one of the options `--system`, `--nodes`, or `--connections`, respectively. In addition, it is now possible to specify from which data node the configuration data is obtained, using the `--config_from_node` option that is added in this release.

For more information, see ndb_config — Extract NDB Cluster Configuration Information. (Bug #11766870)

Bugs Fixed

- **Incompatible Change; NDB Cluster APIs:** Restarting a machine hosting data nodes, SQL nodes, or both, caused such nodes when restarting to time out while trying to obtain node IDs.

  As part of the fix for this issue, the behavior and default values for the NDB API Ndb_cluster_connection::connect() method have been improved. Due to these changes, the version number for the included NDB client library (libndbclient.so) has been increased from 4.0.0 to 5.0.0. For NDB API applications, this means that as part of any upgrade, you must do both of the following:

  - Review and possibly modify any NDB API code that uses the connect() method, in order to take into account its changed default retry handling.
  - Recompile any NDB API applications using the new version of the client library.

  Also in connection with this issue, the default value for each of the two mysqld options --ndb-wait-connected and --ndb-wait-setup has been increased to 30 seconds (from 0 and 15, respectively). In addition, a hard-coded 30-second delay was removed, so that the value of --ndb-wait-connected is now handled correctly in all cases. (Bug #12543299)

- AUTO_INCREMENT values were not set correctly for INSERT IGNORE statements affecting NDB tables. This could lead such statements to fail with Got error 4350 'Transaction already aborted' from NDBCLUSTER when inserting multiple rows containing duplicate values. (Bug #11755237, Bug #46985)

Changes in MySQL NDB Cluster 7.2.0 (5.1.51-ndb-7.2.0) (2011-04-11, Development Milestone Release)

MySQL NDB Cluster 7.2.0 is a development preview release of NDB Cluster, incorporating several new features in the NDB storage engine for testing and user feedback.

Obtaining MySQL NDB Cluster 7.2. MySQL NDB Cluster 7.2.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.1 through MySQL 5.1.51.

Functionality Added or Changed

- **Important Change:** The default values for a number of MySQL NDB Cluster data node configuration parameters have changed, to provide more resiliency to environmental issues and better handling of some potential failure scenarios, and to perform more reliably with increases in memory and other resource requirements brought about by recent improvements in join handling by NDB. The affected parameters are listed here:

  - HeartbeatIntervalDbDb: Default increased from 1500 ms to 5000 ms.
  - ArbitrationTimeout: Default increased from 3000 ms to 7500 ms.
  - TimeBetweenEpochsTimeout: Now effectively disabled by default (default changed from 4000 ms to 0).

- `SharedGlobalMemory`: Default increased from 20MB to 128MB.
- `MaxParallelScansPerFragment`: Default increased from 32 to 256.

In addition, when `MaxNoOfLocalScans` is not specified, the value computed for it automatically has been increased by a factor of 4 (that is, to 4 times `MaxNoOfConcurrentScans`, times the number of data nodes in the cluster).

- **Important Change:** MySQL user privileges can now be distributed automatically across all MySQL servers (SQL nodes) connected to the same MySQL NDB Cluster. Previously, each MySQL Server's user privilege tables were required to use the `MyISAM` storage engine, which meant that a user account and its associated privileges created on one SQL node were not available on any other SQL node without manual intervention. MySQL NDB Cluster now provides an SQL script file `ndb_dist_priv.sql` that can be found in `share/mysql` under the MySQL installation directory; loading this script creates a stored procedure `mysql_cluster_move_privileges` that can be used following initial installation to convert the privilege tables to the `NDB` storage engine, so that any time a MySQL user account is created, dropped, or has its privileges updated on any SQL node, the changes take effect immediately on all other MySQL servers attached to the cluster. Note that you may need to execute `FLUSH PRIVILEGES` on any SQL nodes with connected MySQL clients (or to disconnect and then reconnect the clients) in order for those clients to be able to see the changes in privileges.

  Once `mysql_cluster_move_privileges` has been executed successfully, all MySQL user privileges are distributed across all connected MySQL Servers. MySQL Servers that join the cluster after this automatically participate in the privilege distribution.

  `ndb_dist_priv.sql` also provides stored routines that can be used to verify that the privilege tables have been distributed successfully, and to perform other tasks.

  For more information, see Distributed Privileges Using Shared Grant Tables.

- **Performance:** Added support for pushing down joins to the NDB kernel for parallel execution across the data nodes, which can speed up execution of joins across `NDB` tables by a factor of 20 or more in some cases. Some restrictions apply on the types of joins that can be optimized in this way; in particular, columns to be joined must use exactly the same data type, and cannot be any of the `BLOB` or `TEXT` types. In addition, columns to be joined must be part of a table index or primary key. Support for this feature can be enabled or disabled using the `ndb_join_pushdown` server system variable (enabled by default); see the description of this variable for more information and examples.

  As part of this improvement, the status variables `Ndb_pushed_queries_defined`, `Ndb_pushed_queries_dropped`, `Ndb_pushed_queries_executed`, and `Ndb_pushed_reads` also been introduced for monitoring purposes. You can also see whether a given join is pushed down using `EXPLAIN`. In addition, several new counters relating to push-down join performance have been added to the `counters` table in the `ndbinfo` database. For more information, see the descriptions of the status variables previously mentioned.

Release Series Changelogs: MySQL NDB Cluster 7.2

This section contains unified changelog information for the MySQL NDB Cluster 7.2 release series.

For changelogs covering individual MySQL NDB Cluster 7.2 releases, see NDB Cluster Release Notes.

For general information about features added in MySQL NDB Cluster 7.2, see What is New in MySQL NDB Cluster 7.2.

For an overview of features added in MySQL 5.5 that are not specific to NDB Cluster, see What Is New in MySQL 5.5. For a complete list of all bug fixes and feature changes made in MySQL 5.5 that are not specific to NDB Cluster, see the MySQL 5.5 Release Notes.

Changes in MySQL NDB Cluster 7.2.35 (5.5.62-ndb-7.2.35) (2018-10-23, General availability)

Bugs Fixed

- **Packaging:** The `MySQL-Cluster-devel-gpl` and `MySQL-devel` RPM packages for NDB Cluster did not provide `mysql-devel`. (Bug #66914, Bug #23525339)

Changes in MySQL NDB Cluster 7.2.34 (5.5.61-ndb-7.2.34) (2018-07-27, General availability)

Bugs Fixed

- An internal buffer being reused immediately after it had been freed could lead to an unplanned data node shutdown. (Bug #27622643)

  References: See also: Bug #28698831.

Changes in MySQL NDB Cluster 7.2.33 (5.5.60-ndb-7.2.33) (2018-04-20, General availability)

Bugs Fixed

- Queries using very large lists with `IN` were not handled correctly, which could lead to data node failures. (Bug #27397802)

  References: See also: Bug #28728603.

- `ndb_restore --print-data --hex` did not print trailing 0s of `LONGVARBINARY` values. (Bug #65560, Bug #14198580)

Changes in MySQL NDB Cluster 7.2.32 (5.5.59-ndb-7.2.32) (2018-01-08, General Availability)

Bugs Fixed

- Following `TRUNCATE TABLE` on an `NDB` table, its `AUTO_INCREMENT` ID was not reset on an SQL node not performing binary logging. (Bug #14845851)

- The `NDBFS` block's `OM_SYNC` flag is intended to make sure that all FSWRITEREQ signals used for a given file are synchronized, but was ignored by platforms that do not support `O_SYNC`, meaning that this feature did not behave properly on those platforms. Now the synchronization flag is used on those platforms that do not support `O_SYNC`. (Bug #76975, Bug #21049554)

Changes in MySQL NDB Cluster 7.2.28 (5.5.55-ndb-7.2.28) (2017-04-10, General Availability)

Bugs Fixed

- **NDB Disk Data:** Stale data from NDB Disk Data tables that had been dropped could potentially be included in backups due to the fact that disk scans were enabled for these. To prevent this possibility,

disk scans are now disabled—as are other types of scans—when taking a backup. (Bug #84422, Bug #25353234)

- **NDB Disk Data:** In some cases, setting dynamic in-memory columns of an NDB Disk Data table to `NULL` was not handled correctly. (Bug #79253, Bug #22195588)

Changes in MySQL NDB Cluster 7.2.27 (5.5.54-ndb-7.2.27) (2017-01-17, General Availability)

Bugs Fixed

- A number of potential buffer overflow issues were found and fixed in the `NDB` codebase. (Bug #25260091)

  References: See also: Bug #23152979.

- `ndb_restore` did not restore tables having more than 341 columns correctly. This was due to the fact that the buffer used to hold table metadata read from `.ctl` files was of insufficient size, so that only part of the table descriptor could be read from it in such cases. This issue is fixed by increasing the size of the buffer used by `ndb_restore` for file reads. (Bug #25182956)

  References: See also: Bug #25302901.

Changes in MySQL NDB Cluster 7.2.26 (5.5.53-ndb-7.2.26) (2016-10-18, General Availability)

Bugs Fixed

- 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)

Changes in MySQL NDB Cluster 7.2.25 (5.5.50-ndb-7.2.25) (2016-07-18, General Availability)

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)

Changes in MySQL NDB Cluster 7.2.24 (5.5.48-ndb-7.2.24) (2016-04-19, General Availability)

Bugs Fixed

- 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.

Changes in MySQL NDB Cluster 7.2.23 (5.5.47-ndb-7.2.23) (2016-01-19, General Availability)

Bugs Fixed

- **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)

- 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 NDB 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 NDB 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)

- 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)

- 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)

- `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)

- 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)

Changes in MySQL NDB Cluster 7.2.22 (5.5.46-ndb-7.2.22) (2015-10-19, General Availability)

Bugs Fixed

- **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:** 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)

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

  References: See also: Bug #20204854, 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.

- 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)

- 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)

Changes in MySQL NDB Cluster 7.2.21 (5.5.44-ndb-7.2.21) (2015-07-13, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- 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 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 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

- **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)

- **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)

- 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.

- The multithreaded 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.

- 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)

- 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)

Changes in MySQL NDB Cluster 7.2.20 (5.5.43-ndb-7.2.20) (2015-04-13, 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.

- **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 NDB Cluster 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)

- **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 NDB 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 NDB Cluster 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)

- 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)

- 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)

- 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)

- 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)

- 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 7.2.19 (5.5.41-ndb-7.2.19) (2015-01-25, General Availability)

Bugs Fixed

- **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 Disk Data:** When a node acting as a `DICT` master fails, the arbitrator selects another node to take over in place of the failed node. During the takeover procedure, which includes cleaning up any schema transactions which are still open when the master failed, the disposition of the uncommitted schema transaction is decided. Normally this transaction be rolled back, but if it has completed a sufficient portion of a commit request, the new master finishes processing the commit. Until the fate of the transaction has been decided, no new `TRANS_END_REQ` messages from clients can be processed. In addition, since multiple concurrent schema transactions are not supported, takeover cleanup must be completed before any new transactions can be started.

  A similar restriction applies to any schema operations which are performed in the scope of an open schema transaction. The counter used to coordinate schema operation across all nodes is employed both during takeover processing and when executing any non-local schema operations. This means that starting a schema operation while its schema transaction is in the takeover phase causes this counter to be overwritten by concurrent uses, with unpredictable results.

  The scenarios just described were handled previously using a pseudo-random delay when recovering from a node failure. Now we check before the new master has rolled forward or backwards any schema transactions remaining after the failure of the previous master and avoid starting new schema transactions or performing operations using old transactions until takeover processing has cleaned up after the abandoned transaction. (Bug #19874809, Bug #74503)

- **NDB Disk Data:** When a node acting as `DICT` master fails, it is still possible to request that any open schema transaction be either committed or aborted by sending this request to the new `DICT` master. In this event, the new master takes over the schema transaction and reports back on whether the commit or abort request succeeded. In certain cases, it was possible for the new master to be misidentified—that is, the request was sent to the wrong node, which responded with an error that was interpreted by the client application as an aborted schema transaction, even in cases where the transaction could have been successfully committed, had the correct node been contacted. (Bug #74521, Bug #19880747)

- **NDB Cluster APIs:** The buffer allocated by an `NdbScanOperation` for receiving scanned rows was not released until the `NdbTransaction` owning the scan operation was closed. This could lead to excessive memory usage in an application where multiple scans were created within the same transaction, even if these scans were closed at the end of their lifecycle, unless `NdbScanOperation::close()` was invoked with the *`releaseOp`* argument equal to `true`. Now the buffer is released whenever the cursor navigating the result set is closed with `NdbScanOperation::close()`, regardless of the value of this argument. (Bug #75128, Bug #20166585)

- 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.

- A number of problems relating to the fired triggers pool have been fixed, including the following issues:

  - When the fired triggers pool was exhausted, `NDB` returned Error 218 (`Out of LongMessageBuffer`). A new error code 221 is added to cover this case.

  - An additional, separate case in which Error 218 was wrongly reported now returns the correct error.

  - Setting low values for `MaxNoOfFiredTriggers` led to an error when no memory was allocated if there was only one hash bucket.

  - An aborted transaction now releases any fired trigger records it held. Previously, these records were held until its `ApiConnectRecord` was reused by another transaction.

  - In addition, for the `Fired Triggers` pool in the internal `ndbinfo.ndb$pools` table, the high value always equalled the total, due to the fact that all records were momentarily seized when initializing them. Now the high value shows the maximum following completion of initialization.

  (Bug #19976428)

- Online reorganization when using `ndbmtd` data nodes and with binary logging by `mysqld` enabled could sometimes lead to failures in the `TRIX` and `DBLQH` kernel blocks, or in silent data corruption. (Bug #19903481)

  References: See also: Bug #19912988.

- 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.

- A watchdog failure resulted from a hang while freeing a disk page in `TUP_COMMITREQ`, due to use of an uninitialized block variable. (Bug #19815044, Bug #74380)

- Multiple threads crashing led to multiple sets of trace files being printed and possibly to deadlocks. (Bug #19724313)

- When a client retried against a new master a schema transaction that failed previously against the previous master while the latter was restarting, the lock obtained by this transaction on the new master prevented the previous master from progressing past start phase 3 until the client was terminated, and resources held by it were cleaned up. (Bug #19712569, Bug #74154)

- 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)

- When executing very large pushdown joins involving one or more indexes each defined over several columns, it was possible in some cases for the `DBSPJ` block (see The DBSPJ Block) in the `NDB` kernel to generate `SCAN_FRAGREQ` signals that were excessively large. This caused data nodes to fail when these could not be handled correctly, due to a hard limit in the kernel on the size of such signals (32K). This fix bypasses that limitation by breaking up `SCAN_FRAGREQ` data that is too large for one such signal, and sending the `SCAN_FRAGREQ` as a chunked or fragmented signal instead. (Bug #19390895)

- `ndb_index_stat` sometimes failed when used against a table containing unique indexes. (Bug #18715165)

- Queries against tables containing a CHAR(0) columns failed with `ERROR 1296 (HY000): Got error 4547 'RecordSpecification has overlapping offsets' from NDBCLUSTER`. (Bug #14798022)

- `ndb_restore` failed while restoring a table which contained both a built-in conversion on the primary key and a staging conversion on a `TEXT` column.

  During staging, a `BLOB` table is created with a primary key column of the target type. However, a conversion function was not provided to convert the primary key values before loading them into the staging blob table, which resulted in corrupted primary key values in the staging `BLOB` table. While moving data from the staging table to the target table, the `BLOB` read failed because it could not find the primary key in the `BLOB` table.

  Now all `BLOB` tables are checked to see whether there are conversions on primary keys of their main tables. This check is done after all the main tables are processed, so that conversion functions and parameters have already been set for the main tables. Any conversion functions and parameters used for the primary key in the main table are now duplicated in the `BLOB` table. (Bug #73966, Bug #19642978)

- 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)

- `ndb_restore --print-data` truncated `TEXT` and `BLOB` column values to 240 bytes rather than 256 bytes. (Bug #65467, Bug #14571512)

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

Changes in MySQL NDB Cluster 7.2.18 (5.5.40-ndb-7.2.18) (2014-10-16, General Availability)

Functionality Added or Changed

- Added the `--exclude-missing-tables` option for `ndb_restore`. When enabled, the option causes tables present in the backup but not in the target database to be ignored. (Bug #57566, Bug #11764704)

Bugs Fixed

- **NDB Cluster APIs:** The fix for Bug #16723708 stopped the `ndb_logevent_get_next()` function from casting a log event's `ndb_mgm_event_category` to an `enum` type, but this change interfered with existing applications, and so the function's original behavior is now reinstated. A new MGM API function exhibiting the corrected behavior `ndb_logevent_get_next2()` has been added in this release to take the place of the reverted function, for use in applications that do not require backward compatibility. In all other respects apart from this, the new function is identical with its predecessor. (Bug #18354165)

  References: Reverted patches: Bug #16723708.

- **NDB Cluster APIs:** NDB API scans leaked `Ndb_cluster_connection` objects after `nextResult()` was called when an operation resulted in an error. This leak locked up the corresponding connection objects in the `DBTC` kernel block until the connection was closed. (Bug #17730825, Bug #20170731)

- When assembling error messages of the form `Incorrect state for node` *`n`* `state:` *`node_state`*, written when the transporter failed to connect, the node state was used in place of the node ID in a number of instances, which resulted in errors of this type for which the node state was reported incorrectly. (Bug #19559313, Bug #73801)

- 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)

- The failure of a data node could in some situations cause a set of API nodes to fail as well due to the sending of a `CLOSE_COMREQ` signal that was sometimes not completely initialized. (Bug #19513967)

- A more detailed error report is printed in the event of a critical failure in one of the `NDB` internal `sendSignal*()` methods, prior to crashing the process, as was already implemented for `sendSignal()`, but was missing from the more specialized `sendSignalNoRelease()` method. Having a crash of this type correctly reported can help with identifying configuration hardware issues in some cases. (Bug #19414511)

  References: See also: Bug #19390895.

- `ndb_restore` failed to restore the cluster's metadata when there were more than approximately 17 K data objects. (Bug #19202654)

- 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.

- Parallel transactions performing reads immediately preceding a delete on the same tuple could cause the NDB kernel to crash. This was more likely to occur when separate TC threads were specified using the ThreadConfig configuration parameter. (Bug #19031389)

- Attribute promotion between different TEXT types (any of TINYTEXT, TEXT, MEDIUMTEXT, and LONGTEXT) by ndb_restore was not handled properly in some cases. In addition, TEXT values are now truncated according to the limits set by mysqld (for example, values converted to TINYTEXT from another type are truncated to 256 bytes). In the case of columns using a multibyte character set, the value is truncated to the end of the last well-formed character.

  Also as a result of this fix, conversion to a TEXT column of any size that uses a different character set from the original is now disallowed. (Bug #18875137)

- To assist with diagnostic issues where many watchdog warnings are raised, it is now possible to activate (or deactivate) a killer watchdog using DUMP 2610 in the ndb_mgm client. When set, this shuts down the data node on which the next watchdog warning occurs, providing a trace log. (Bug #18703922)

- The NDB optimized node recovery mechanism attempts to transfer only relevant page changes to a starting node in order to speed the recovery process; this is done by having the starting node indicate the index of the last global checkpoint (GCI) in which it participated, so that the node that was already running copies only data for rows which have changed since that GCI. Every row has a GCI metacolumn which facilitates this; for a deleted row, the slot formerly stpring this row's data contains a GCI value, and for deleted pages, every row on the missing page is considered changed and thus needs to be sent.

  When these changes are received by the starting node, this node performs a lookup for the page and index to determine what they contain. This lookup could cause a real underlying page to be mapped against the logical page ID, even when this page contained no data.

  One way in which this issue could manifest itself occurred after cluster DataMemory usage approached maximum, and deletion of many rows followed by a rolling restart of the data nodes was performed with the expectation that this would free memory, but in fact it was possible in this scenario for memory not to be freed and in some cases for memory usage actually to increase to its maximum.

  This fix solves these issues by ensuring that a real physical page is mapped to a logical ID during node recovery only when this page contains actual data which needs to be stored. (Bug #18683398, Bug #18731008)

- mysqld failed while attempting a read removal before a delete with an index merge. This occurred only when a quick select was also generated for the delete operation. During read removal, the index from the quick select is used to access the table structure. In this case, because the delete uses an index merge, the quick select index is set to MAX_KEY instead of a valid index value, which led to a bad pointer (which was then dereferenced).

  The fix for this problem adds a check so that read removal before delete is not attempted if the quick select index has a value of MAX_KEY. (Bug #18487960)

- When a data node sent a MISSING_DATA signal due to a buffer overflow and no event data had yet been sent for the current epoch, the dummy event list created to handle this inconsistency was not deleted after the information in the dummy event list was transferred to the completed list. (Bug #18410939)

- Incorrect calculation of the next autoincrement value following a manual insertion towards the end of a cached range could result in duplicate values sometimes being used. This issue could

manifest itself when using certain combinations of values for `auto_increment_increment`, `auto_increment_offset`, and `ndb_autoincrement_prefetch_sz`.

This issue has been fixed by modifying the calculation to make sure that the next value from the cache as computed by `NDB` is of the form `auto_increment_offset + (N * auto_increment_increment`. This avoids any rounding up by the MySQL Server of the returned value, which could result in duplicate entries when the rounded-up value fell outside the range of values cached by `NDB`. (Bug #17893872)

- `ndb_show_tables --help` output contained misleading information about the `--database` (`-d`) option. In addition, the long form of the option (`--database`) did not work properly. (Bug #17703874)
- Using the `--help` option with `ndb_print_file` caused the program to segfault. (Bug #17069285)
- 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)

Changes in MySQL NDB Cluster 7.2.17 (5.5.37-ndb-7.2.17) (2014-07-11, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **NDB Cluster APIs:** Added as an aid to debugging the ability to specify a human-readable name for a given `Ndb` object and later to retrieve it. These operations are implemented, respectively, as the `setNdbObjectName()` and `getNdbObjectName()` methods.

  To make tracing of event handling between a user application and `NDB` easier, you can use the reference (from `getReference()` followed by the name (if provided) in printouts; the reference ties together the application `Ndb` object, the event buffer, and the `NDB` storage engine's `SUMA` block. (Bug #18419907)

Bugs Fixed

- **NDB Cluster APIs:** When an `NDB` data node indicates a buffer overflow via an empty epoch, the event buffer places an inconsistent data event in the event queue. When this was consumed, it was not removed from the event queue as expected, causing subsequent `nextEvent()` calls to return 0. This caused event consumption to stall because the inconsistency remained flagged forever, while event data accumulated in the queue.

  Event data belonging to an empty inconsistent epoch can be found either at the beginning or somewhere in the middle. `pollEvents()` returns 0 for the first case. This fix handles the second case: calling `nextEvent()` call dequeues the inconsistent event before it returns. In order to benefit from this fix, user applications must call `nextEvent()` even when `pollEvents()` returns 0. (Bug #18716991)

- **NDB Cluster APIs:** The `pollEvents()` method returned 1, even when called with a wait time equal to 0, and there were no events waiting in the queue. Now in such cases it returns 0 as expected. (Bug #18703871)
- Processing a NODE_FAILREP signal that contained an invalid node ID could cause a data node to fail. (Bug #18993037, Bug #73015)

References: This issue is a regression of: Bug #16007980.

- When building out of source, some files were written to the source directory instead of the build dir. These included the `manifest.mf` files used for creating ClusterJ jars and the `pom.xml` file used by `mvn_install_ndbjtie.sh`. In addition, `ndbinfo.sql` was written to the build directory, but marked as output to the source directory in `CMakeLists.txt`. (Bug #18889568, Bug #72843)

- It was possible for a data node restart to become stuck indefinitely in start phase 101 (see Summary of NDB Cluster Start Phases) when there were connection problems between the node being restarted and one or more subscribing API nodes.

  To help prevent this from happening, a new data node configuration parameter `RestartSubscriberConnectTimeout` has been introduced, which can be used to control how long a data node restart can stall in start phase 101 before giving up and attempting to restart again. The default is 12000 ms. (Bug #18599198)

- 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.

- Following a long series of inserts, when running with a relatively small redo log and an insufficient large value for `MaxNoOfConcurrentTransactions`, there remained transactions that were blocked by the lack of redo log and were thus not aborted in the correct state (waiting for prepare log to be sent to disk, or `LOG_QUEUED` state). This caused the redo log to remain blocked until unblocked by a completion of a local checkpoint. This could lead to a deadlock, when the blocked aborts in turned blocked global checkpoints, and blocked GCPs block LCPs. To prevent this situation from arising, we now abort immediately when we reach the `LOG_QUEUED` state in the abort state handler. (Bug #18533982)

- `ndbmtd` supports multiple parallel receiver threads, each of which performs signal reception for a subset of the remote node connections (transporters) with the mapping of remote_nodes to receiver threads decided at node startup. Connection control is managed by the multi-instance `TRPMAN` block, which is organized as a proxy and workers, and each receiver thread has a `TRPMAN` worker running locally.

  The `QMGR` block sends signals to `TRPMAN` to enable and disable communications with remote nodes. These signals are sent to the `TRPMAN` proxy, which forwards them to the workers. The workers themselves decide whether to act on signals, based on the set of remote nodes they manage.

  The current issue arises because the mechanism used by the `TRPMAN` workers for determining which connections they are responsible for was implemented in such a way that each worker thought it was responsible for all connections. This resulted in the `TRPMAN` actions for `OPEN_COMORD`, `ENABLE_COMREQ`, and `CLOSE_COMREQ` being processed multiple times.

  The fix keeps `TRPMAN` instances (receiver threads) executing `OPEN_COMORD`, `ENABLE_COMREQ` and `CLOSE_COMREQ` requests. In addition, the correct `TRPMAN` instance is now chosen when routing from this instance for a specific remote connection. (Bug #18518037)

- During data node failure handling, the transaction coordinator performing takeover gathers all known state information for any failed TC instance transactions, determines whether each transaction has been committed or aborted, and informs any involved API nodes so that they can report this accurately to their clients. The TC instance provides this information by sending `TCKEY_FAILREF` or `TCKEY_FAILCONF` signals to the API nodes as appropriate top each affected transaction.

In the event that this TC instance does not have a direct connection to the API node, it attempts to deliver the signal by routing it through another data node in the same node group as the failing TC, and sends a GSN_TCKEY_FAILREFCONF_R signal to TC block instance 0 in that data node. A problem arose in the case of multiple transaction cooridnators, when this TC instance did not have a signal handler for such signals, which led it to fail.

This issue has been corrected by adding a handler to the TC proxy block which in such cases forwards the signal to one of the local TC worker instances, which in turn attempts to forward the signal on to the API node. (Bug #18455971)

- When running with a very slow main thread, and one or more transaction coordinator threads, on different CPUs, it was possible to encounter a timeout when sending a DIH_SCAN_GET_NODESREQ signal, which could lead to a crash of the data node. Now in such cases the timeout is avoided. (Bug #18449222)

- 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)

- A local checkpoint (LCP) is tracked using a global LCP state (c_lcpState), and each NDB table has a status indicator which indicates the LCP status of that table (tabLcpStatus). If the global LCP state is LCP_STATUS_IDLE, then all the tables should have an LCP status of TLS_COMPLETED.

  When an LCP starts, the global LCP status is LCP_INIT_TABLES and the thread starts setting all the NDB tables to TLS_ACTIVE. If any tables are not ready for LCP, the LCP initialization procedure continues with CONTINUEB signals until all tables have become available and been marked TLS_ACTIVE. When this initialization is complete, the global LCP status is set to LCP_STATUS_ACTIVE.

  This bug occurred when the following conditions were met:

  - An LCP was in the LCP_INIT_TABLES state, and some but not all tables had been set to TLS_ACTIVE.

  - The master node failed before the global LCP state changed to LCP_STATUS_ACTIVE; that is, before the LCP could finish processing all tables.

  - The NODE_FAILREP signal resulting from the node failure was processed before the final CONTINUEB signal from the LCP initialization process, so that the node failure was processed while the LCP remained in the LCP_INIT_TABLES state.

  Following master node failure and selection of a new one, the new master queries the remaining nodes with a MASTER_LCPREQ signal to determine the state of the LCP. At this point, since the LCP status was LCP_INIT_TABLES, the LCP status was reset to LCP_STATUS_IDLE. However, the LCP status of the tables was not modified, so there remained tables with TLS_ACTIVE. Afterwards, the failed node is removed from the LCP. If the LCP status of a given table is TLS_ACTIVE, there is a check that the global LCP status is not LCP_STATUS_IDLE; this check failed and caused the data node to fail.

  Now the MASTER_LCPREQ handler ensures that the tabLcpStatus for all tables is updated to TLS_COMPLETED when the global LCP status is changed to LCP_STATUS_IDLE. (Bug #18044717)

- When performing a copying ALTER TABLE operation, mysqld creates a new copy of the table to be altered. This intermediate table, which is given a name bearing the prefix #sql-, has an updated schema but contains no data. mysqld then copies the data from the original table to this intermediate

table, drops the original table, and finally renames the intermediate table with the name of the original table.

`mysqld` regards such a table as a temporary table and does not include it in the output from `SHOW TABLES`; `mysqldump` also ignores an intermediate table. However, `NDB` sees no difference between such an intermediate table and any other table. This difference in how intermediate tables are viewed by `mysqld` (and MySQL client programs) and by the `NDB` storage engine can give rise to problems when performing a backup and restore if an intermediate table existed in `NDB`, possibly left over from a failed `ALTER TABLE` that used copying. If a schema backup is performed using `mysqldump` and the `mysql` client, this table is not included. However, in the case where a data backup was done using the `ndb_mgm` client's `BACKUP` command, the intermediate table was included, and was also included by `ndb_restore`, which then failed due to attempting to load data into a table which was not defined in the backed up schema.

To prevent such failures from occurring, `ndb_restore` now by default ignores intermediate tables created during `ALTER TABLE` operations (that is, tables whose names begin with the prefix `#sql-`). A new option `--exclude-intermediate-sql-tables` is added that makes it possible to override the new behavior. The option's default value is `TRUE`; to cause `ndb_restore` to revert to the old behavior and to attempt to restore intermediate tables, set this option to `FALSE`. (Bug #17882305)

- The logging of insert failures has been improved. This is intended to help diagnose occasional issues seen when writing to the `mysql.ndb_binlog_index` table. (Bug #17461625)
- Employing a `CHAR` column that used the `UTF8` character set as a table's primary key column led to node failure when restarting data nodes. Attempting to restore a table with such a primary key also caused `ndb_restore` to fail. (Bug #16895311, Bug #68893)
- The `--order` (`-o`) option for the `ndb_select_all` utility worked only when specified as the last option, and did not work with an equals sign.

  As part of this fix, the program's `--help` output was also aligned with the `--order` option's correct behavior. (Bug #64426, Bug #16374870)

Changes in MySQL NDB Cluster 7.2.16 (5.5.37-ndb-7.2.16) (2014-04-08, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Handling of `LongMessageBuffer` shortages and statistics has been improved as follows:
  - The default value of `LongMessageBuffer` has been increased from 4 MB to 64 MB.
  - When this resource is exhausted, a suitable informative message is now printed in the data node log describing possible causes of the problem and suggesting possible solutions.
  - `LongMessageBuffer` usage information is now shown in the `ndbinfo.memoryusage` table. See the description of this table for an example and additional information.

Bugs Fixed

- **Important Change:** The server system variables `ndb_index_cache_entries` and `ndb_index_stat_freq`, which had been deprecated in a previous MySQL NDB Cluster release series, have now been removed. (Bug #11746486, Bug #26673)

- **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)

- **NDB Cluster APIs:** Refactoring that was performed in MySQL NDB Cluster 7.2.15 inadvertently introduced a dependency in `Ndb.hpp` on a file that is not included in the distribution, which caused NDB API applications to fail to compile. The dependency has been removed. (Bug #18293112, Bug #71803)

  References: This issue is a regression of: Bug #17647637.

- **NDB Cluster APIs:** An NDB API application sends a scan query to a data node; the scan is processed by the transaction coordinator (TC). The TC forwards a `LQHKEYREQ` request to the appropriate LDM, and aborts the transaction if it does not receive a `LQHKEYCONF` response within the specified time limit. After the transaction is successfully aborted, the TC sends a `TCROLLBACKREP` to the NDBAPI client, and the NDB API client processes this message by cleaning up any `Ndb` objects associated with the transaction.

  The client receives the data which it has requested in the form of `TRANSID_AI` signals, buffered for sending at the data node, and may be delivered after a delay. On receiving such a signal, `NDB` checks the transaction state and ID: if these are as expected, it processes the signal using the `Ndb` objects associated with that transaction.

  The current bug occurs when all the following conditions are fulfilled:

  - The transaction coordinator aborts a transaction due to delays and sends a `TCROLLBACPREP` signal to the client, while at the same time a `TRANSID_AI` which has been buffered for delivery at an LDM is delivered to the same client.
  - The NDB API client considers the transaction complete on receipt of a `TCROLLBACKREP` signal, and immediately closes the transaction.
  - The client has a separate receiver thread running concurrently with the thread that is engaged in closing the transaction.
  - The arrival of the late `TRANSID_AI` interleaves with the closing of the user thread's transaction such that `TRANSID_AI` processing passes normal checks before `closeTransaction()` resets the transaction state and invalidates the receiver.

  When these conditions are all met, the receiver thread proceeds to continue working on the `TRANSID_AI` signal using the invalidated receiver. Since the receiver is already invalidated, its usage results in a node failure.

  Now the `Ndb` object cleanup done for `TCROLLBACKREP` includes invalidation of the transaction ID, so that, for a given transaction, any signal which is received after the `TCROLLBACKREP` arrives does not pass the transaction ID check and is silently dropped. This fix is also implemented for the `TC_COMMITREF`, `TCROLLBACKREF`, `TCKEY_FAILCONF`, and `TCKEY_FAILREF` signals as well.

  See also Operations and Signals, for additional information about NDB messaging. (Bug #18196562)

- **NDB Cluster APIs:** The example `ndbapi-examples/ndbapi_blob_ndbrecord/main.cpp` included an internal header file (`ndb_global.h`) not found in the MySQL NDB Cluster binary distribution. The example now uses `stdlib.h` and `string.h` instead of this file. (Bug #18096866, Bug #71409)

- **NDB Cluster APIs:** `ndb_restore` could sometimes report `Error 701 System busy with other schema operation` unnecessarily when restoring in parallel. (Bug #17916243)

- When an `ALTER TABLE` statement changed table schemas without causing a change in the table's partitioning, the new table definition did not copy the hash map from the old definition, but used the current default hash map instead. However, the table data was not reorganized according to the new hashmap, which made some rows inaccessible using a primary key lookup if the two hash maps had incompatible definitions.

  To keep this situation from occurring, any `ALTER TABLE` that entails a hashmap change now triggers a reorganisation of the table. In addition, when copying a table definition in such cases, the hashmap is now also copied. (Bug #18436558)

- 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)

- Checking of timeouts is handled by the signal `TIME_SIGNAL`. Previously, this signal was generated by the `QMGR` `NDB` kernel block in the main thread, and sent to the `QMRG`, `DBLQH`, and `DBTC` blocks (see NDB Kernel Blocks) as needed to check (respectively) heartbeats, disk writes, and transaction timeouts. In `ndbmtd` (as opposed to `ndbd`), these blocks all execute in different threads. This meant that if, for example, `QMGR` was actively working and some other thread was put to sleep, the previously sleeping thread received a large number of `TIME_SIGNAL` messages simultaneously when it was woken up again, with the effect that effective times moved very quickly in `DBLQH` as well as in `DBTC`. In `DBLQH`, this had no noticeable adverse effects, but this was not the case in `DBTC`; the latter block could not work on transactions even though time was still advancing, leading to a situation in which many operations appeared to time out because the transaction coordinator (TC) thread was comparatively slow in answering requests.

  In addition, when the TC thread slept for longer than 1500 milliseconds, the data node crashed due to detecting that the timeout handling loop had not yet stopped. To rectify this problem, the generation of the `TIME_SIGNAL` has been moved into the local threads instead of `QMGR`; this provides for better control over how quickly `TIME_SIGNAL` messages are allowed to arrive. (Bug #18417623)

- The `ServerPort` and `TcpBind_INADDR_ANY` configuration parameters were not included in the output of `ndb_mgmd --print-full-config`. (Bug #18366909)

- After dropping an `NDB` table, neither the cluster log nor the output of the `REPORT MemoryUsage` command showed that the `IndexMemory` used by that table had been freed, even though the memory had in fact been deallocated. This issue was introduced in MySQL NDB Cluster 7.2.14. (Bug #18296810)

- `ndb_show_tables` sometimes failed with the error message `Unable to connect to management server` and immediately terminated, without providing the underlying reason for the failure. To provide more useful information in such cases, this program now also prints the most recent error from the `Ndb_cluster_connection` object used to instantiate the connection. (Bug #18276327)

- `-DWITH_NDBMTD=0` did not function correctly, which could cause the build to fail on platforms such as ARM and Raspberry Pi which do not define the memory barrier functions required to compile `ndbmtd`. (Bug #18267919)

  References: See also: Bug #16620938.

- The block threads managed by the multithreading scheduler communicate by placing signals in an out queue or job buffer which is set up between all block threads. This queue has a fixed maximum size, such that when it is filled up, the worker thread must wait for the consumer to drain the queue. In a highly loaded system, multiple threads could end up in a circular wait lock due to full out buffers, such that they

were preventing each other from performing any useful work. This condition eventually led to the data node being declared dead and killed by the watchdog timer.

To fix this problem, we detect situations in which a circular wait lock is about to begin, and cause buffers which are otherwise held in reserve to become available for signal processing by queues which are highly loaded. (Bug #18229003)

- The `ndb_mgm` client `START BACKUP` command (see Commands in the NDB Cluster Management Client) could experience occasional random failures when a ping was received prior to an expected `BackupCompleted` event. Now the connection established by this command is not checked until it has been properly set up. (Bug #18165088)

- The local checkpoint lag watchdog tracking the number of times a check for LCP timeout was performed using the system scheduler and used this count to check for a timeout condition, but this caused a number of issues. To overcome these limitations, the LCP watchdog has been refactored to keep track of its own start times, and to calculate elapsed time by reading the (real) clock every time it is called. (Bug #17842035)

  References: See also: Bug #17647469.

- Data nodes running `ndbmtd` could stall while performing an online upgrade of a MySQL NDB Cluster containing a great many tables from a version prior to NDB 7.2.5 to version 7.2.5 or later. (Bug #16693068)

Changes in MySQL NDB Cluster 7.2.15 (5.5.35-ndb-7.2.15) (2014-02-05, General Availability)

Bugs Fixed

- **Packaging; Solaris:** Compilation of `ndbmtd` failed on Solaris 10 and 11 for 32-bit `x86`, and the binary was not included in the binary distributions for these platforms. (Bug #16620938)

- **Microsoft Windows:** Timers used in timing scheduler events in the `NDB` kernel have been refactored, in part to insure that they are monotonic on all platforms. In particular, on Windows, event intervals were previously calculated using values obtained from `GetSystemTimeAsFileTime()`, which reads directly from the system time ("wall clock"), and which may arbitrarily be reset backward or forward, leading to false watchdog or heartbeat alarms, or even node shutdown. Lack of timer monotonicity could also cause slow disk writes during backups and global checkpoints. To fix this issue, the Windows implementation now uses `QueryPerformanceCounters()` instead of `GetSystemTimeAsFileTime()`. In the event that a monotonic timer is not found on startup of the data nodes, a warning is logged.

  In addition, on all platforms, a check is now performed at compile time for available system monotonic timers, and the build fails if one cannot be found; note that `CLOCK_HIGHRES` is now supported as an alternative for `CLOCK_MONOTONIC` if the latter is not available. (Bug #17647637)

- **NDB Disk Data:** When using Disk Data tables and `ndbmtd` data nodes, it was possible for the undo buffer to become overloaded, leading to a crash of the data nodes. This issue was more likely to be encountered when using Disk Data columns whose size was approximately 8K or larger. (Bug #16766493)

- **NDB Cluster APIs:** `UINT_MAX64` was treated as a signed value by Visual Studio 2010. To prevent this from happening, the value is now explicitly defined as unsigned. (Bug #17947674)

  References: See also: Bug #17647637.

- **NDB Cluster APIs:** It was possible for an `Ndb` object to receive signals for handling before it was initialized, leading to thread interleaving and possible data node failure when executing a call to `Ndb::init()`. To guard against this happening, a check is now made when it is starting to receive signals that the `Ndb` object is properly initialized before any signals are actually handled. (Bug #17719439)

- **NDB Cluster APIs:** Compilation of example NDB API program files failed due to missing include directives. (Bug #17672846, Bug #70759)

- Monotonic timers on several platforms can experience issues which might result in the monotonic clock doing small jumps back in time. This is due to imperfect synchronization of clocks between multiple CPU cores and does not normally have an adverse effect on the scheduler and watchdog mechanisms; so we handle some of these cases by making backtick protection less strict, although we continue to ensure that the backtick is less than 10 milliseconds. This fix also removes several checks for backticks which are thereby made redundant. (Bug #17973819)

- Under certain specific circumstances, in a cluster having two SQL nodes, one of these could hang, and could not be accessed again even after killing the `mysqld` process and restarting it. (Bug #17875885, Bug #18080104)

  References: See also: Bug #17934985.

- Poor support or lack of support on some platforms for monotonic timers caused issues with delayed signal handling by the job scheduler for the multithreaded data node. Variances (timer leaps) on such platforms are now handled in the same way the multithreaded data node process that they are by the singlethreaded version. (Bug #17857442)

  References: See also: Bug #17475425, Bug #17647637.

- In some cases, with `ndb_join_pushdown` enabled, it was possible to obtain from a valid query the error `Got error 290 'Corrupt key in TC, unable to xfrm' from NDBCLUSTER` even though the data was not actually corrupted.

  It was determined that a `NULL` in a `VARCHAR` column could be used to construct a lookup key, but since `NULL` is never equal to any other value, such a lookup could simple have been eliminated instead. This `NULL` lookup in turn led to the spurious error message.

  This fix takes advantage of the fact that a key lookup with `NULL` never finds any matching rows, and so `NDB` does not try to perform the lookup that would have led to the error. (Bug #17845161)

- It was theoretically possible in certain cases for a number of output functions internal to the `NDB` code to supply an uninitialized buffer as output. Now in such cases, a newline character is printed instead. (Bug #17775602, Bug #17775772)

- Use of the `localtime()` function in `NDB` multithreading code led to otherwise nondeterministic failures in `ndbmtd`. This fix replaces this function, which on many platforms uses a buffer shared among multiple threads, with `localtime_r()`, which can have allocated to it a buffer of its own. (Bug #17750252)

- When using single-threaded (`ndbd`) data nodes with `RealTimeScheduler` enabled, the CPU did not, as intended, temporarily lower its scheduling priority to normal every 10 milliseconds to give other, non-realtime threads a chance to run. (Bug #17739131)

- During arbitrator selection, `QMGR` (see The QMGR Block) runs through a series of states, the first few of which are (in order) `NULL`, `INIT`, `FIND`, `PREP1`, `PREP2`, and `START`. A check for an arbitration selection timeout occurred in the `FIND` state, even though the corresponding timer was not set until `QMGR` reached the `PREP1` and `PREP2` states. Attempting to read the resulting uninitialized timestamp value could lead to false `Could not find an arbitrator, cluster is not partition-safe` warnings.

This fix moves the setting of the timer for arbitration timeout to the `INIT` state, so that the value later read during `FIND` is always initialized. (Bug #17738720)

- The global checkpoint lag watchdog tracking the number of times a check for GCP lag was performed using the system scheduler and used this count to check for a timeout condition, but this caused a number of issues. To overcome these limitations, the GCP watchdog has been refactored to keep track of its own start times, and to calculate elapsed time by reading the (real) clock every time it is called.

  In addition, any backticks (rare in any case) are now handled by taking the backward time as the new current time and calculating the elapsed time for this round as 0. Finally, any ill effects of a forward leap, which possibly could expire the watchdog timer immediately, are reduced by never calculating an elapsed time longer than the requested delay time for the watchdog timer. (Bug #17647469)

  References: See also: Bug #17842035.

- The length of the interval (intended to be 10 seconds) between warnings for `GCP_COMMIT` when the GCP progress watchdog did not detect progress in a global checkpoint was not always calculated correctly. (Bug #17647213)

- In certain rare cases on commit of a transaction, an `Ndb` object was released before the transaction coordinator (`DBTC` kernel block) sent the expected `COMMIT_CONF` signal; `NDB` failed to send a `COMMIT_ACK` signal in response, which caused a memory leak in the `NDB` kernel could later lead to node failure.

  Now an `Ndb` object is not released until the `COMMIT_CONF` signal has actually been received. (Bug #16944817)

- 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 #16890703)

  References: See also: Bug #21184102.

- Losing its connections to the management node or data nodes while a query against the `ndbinfo.memoryusage` table was in progress caused the SQL node where the query was issued to fail. (Bug #14483440, Bug #16810415)

- The `ndbd_redo_log_reader` utility now supports a `--help` option. Using this options causes the program to print basic usage information, and then to exit. (Bug #11749591, Bug #36805)

Changes in MySQL NDB Cluster 7.2.14 (5.5.34-ndb-7.2.14) (2013-10-31, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Added the `ExtraSendBufferMemory` parameter for management nodes and API nodes. (Formerly, this parameter was available only for configuring data nodes.) See `ExtraSendBufferMemory` (management nodes), and `ExtraSendBufferMemory` (API nodes), for more information. (Bug #14555359)

- `BLOB` and `TEXT` columns are now reorganized by the `ALTER ONLINE TABLE ... REORGANIZE PARTITION` statement. (Bug #13714148)

Bugs Fixed

- **Performance:** In a number of cases found in various locations in the MySQL NDB Cluster codebase, unnecessary iterations were performed; this was caused by failing to break out of a repeating control structure after a test condition had been met. This community-contributed fix removes the unneeded repetitions by supplying the missing breaks. (Bug #16904243, Bug #69392, Bug #16904338, Bug #69394, Bug #16778417, Bug #69171, Bug #16778494, Bug #69172, Bug #16798410, Bug #69207, Bug #16801489, Bug #69215, Bug #16904266, Bug #69393)

- **Packaging; Microsoft Windows:** The MySQL NDB Cluster installer for Windows provided a nonfunctional option to install debug symbols (contained in `*.pdb` files). This option has been removed from the installer.

  

  Note

  You can obtain the `*.pdb` debug files for a given MySQL NDB Cluster release from the Windows `.zip` archive for the same release, such as `mysql-cluster-gpl-7.2.14-win32.zip` or `mysql-cluster-gpl-7.3.2-winx64.zip`.

  (Bug #16748308, Bug #69112)

- **Packaging; Microsoft Windows:** The MySQL NDB Cluster Windows installer attempted to use the wrong path to the `my.ini` file and the executables directory. (Bug #13813120, Bug #64510)

- **Packaging:** A fix has been made in this release correcting a number of issues found in some recent `MySQL-Cluster-server` RPM packages, including dependencies on the `mysql-server` package and conflicts with the `mysql-libs` package needed by other common applications. Platforms known to have been affected by these issues include CentOS 6, Red Hat Enterprise Linux 6, and Oracle Linux 6. (Bug #14063590, Bug #14181419, Bug #65534)

- **Microsoft Windows:** The Windows error *`ERROR_FILE_EXISTS`* was not recognized by `NDB`, which treated it as an unknown error. (Bug #16970960)

- **NDB Disk Data:** The statements `CREATE TABLESPACE`, `ALTER LOGFILE GROUP`, and `ALTER TABLESPACE` failed with a syntax error when `INITIAL_SIZE` was specified using letter abbreviations such as `M` or `G`. In addition, `CREATE LOGFILE GROUP` failed when `INITIAL_SIZE`, `UNDO_BUFFER_SIZE`, or both options were specified using letter abbreviations. (Bug #13116514, Bug #16104705, Bug #62858)

- **NDB Cluster APIs:** For each log event retrieved using the MGM API, the log event category (`ndb_mgm_event_category`) was simply cast to an `enum` type, which resulted in invalid category values. Now an offset is added to the category following the cast to ensure that the value does not fall out of the allowed range.

  

  Note

  This change was reverted by the fix for Bug #18354165. See the MySQL NDB Cluster API Developer documentation for `ndb_logevent_get_next()`, for more information.

  (Bug #16723708)

  References: See also: Bug #18354165.

- **NDB Cluster APIs:** The `Event::setTable()` method now supports a pointer or a reference to table as its required argument. If a null table pointer is used, the method now returns -1 to make it clear that this is what has occurred. (Bug #16329082)
- Trying to restore to a table having a `BLOB` column in a different position from that of the original one caused `ndb_restore --restore-data` to fail. (Bug #17395298)
- `ndb_restore` could abort during the last stages of a restore using attribute promotion or demotion into an existing table. This could happen if a converted attribute was nullable and the backup had been run on active database. (Bug #17275798)
- The `DBUTIL` data node block is now less strict about the order in which it receives certain messages from other nodes. (Bug #17052422)
- `RealTimeScheduler` did not work correctly with data nodes running `ndbmtd`. (Bug #16961971)
- File system errors occurring during a local checkpoint could sometimes cause an LCP to hang with no obvious cause when they were not handled correctly. Now in such cases, such errors always cause the node to fail. Note that the LQH block always shuts down the node when a local checkpoint fails; the change here is to make likely node failure occur more quickly and to make the original file system error more visible. (Bug #16961443)
- `mysql_upgrade` failed when upgrading from MySQL NDB Cluster 7.1.26 to MySQL NDB Cluster 7.2.13 when it attempted to invoke a stored procedure before the `mysql.proc` table had been upgraded. (Bug #16933405)

  References: This issue is a regression of: Bug #16226274.
- The planned or unplanned shutdown of one or more data nodes while reading table data from the `ndbinfo` database caused a memory leak. (Bug #16932989)
- Maintenance and checking of parent batch completion in the `SPJ` block of the `NDB` kernel was reimplemented. Among other improvements, the completion state of all ancestor nodes in the tree are now preserved. (Bug #16925513)
- Executing `DROP TABLE` while `DBDIH` was updating table checkpoint information subsequent to a node failure could lead to a data node failure. (Bug #16904469)
- In certain cases, when starting a new SQL node, `mysqld` failed with Error 1427 `Api node died, when SUB_START_REQ reached node`. (Bug #16840741)
- Failure to use container classes specific `NDB` during node failure handling could cause leakage of commit-ack markers, which could later lead to resource shortages or additional node crashes. (Bug #16834416)
- Use of an uninitialized variable employed in connection with error handling in the `DBLQH` kernel block could sometimes lead to a data node crash or other stability issues for no apparent reason. (Bug #16834333)
- A race condition in the time between the reception of a `execNODE_FAILREP` signal by the `QMGR` kernel block and its reception by the `DBLQH` and `DBTC` kernel blocks could lead to data node crashes during shutdown. (Bug #16834242)
- The `CLUSTERLOG` command (see Commands in the NDB Cluster Management Client) caused `ndb_mgm` to crash on Solaris SPARC systems. (Bug #16834030)
- The NDB Error-Reporting Utility (`ndb_error_reporter`) failed to include the cluster nodes' log files in the archive it produced when the `FILE` option was set for the parameter `LogDestination`. (Bug #16765651)

References: See also: Bug #11752792, Bug #44082.

- The LCP fragment scan watchdog periodically checks for lack of progress in a fragment scan performed as part of a local checkpoint, and shuts down the node if there is no progress after a given amount of time has elapsed. This interval, formerly hard-coded as 60 seconds, can now be configured using the `LcpScanProgressTimeout` data node configuration parameter added in this release.

  This configuration parameter sets the maximum time the local checkpoint can be stalled before the LCP fragment scan watchdog shuts down the node. The default is 60 seconds, which provides backward compatibility with previous releases.

  You can disable the LCP fragment scan watchdog by setting this parameter to 0. (Bug #16630410)

- Added the `ndb_error_reporter` options `--connection-timeout`, which makes it possible to set a timeout for connecting to nodes, `--dry-scp`, which disables scp connections to remote hosts, and `--skip-nodegroup`, which skips all nodes in a given node group. (Bug #16602002)

  References: See also: Bug #11752792, Bug #44082.

- After issuing `START BACKUP id WAIT STARTED`, if *`id`* had already been used for a backup ID, an error caused by the duplicate ID occurred as expected, but following this, the `START BACKUP` command never completed. (Bug #16593604, Bug #68854)

- `ndb_mgm` treated backup IDs provided to `ABORT BACKUP` commands as signed values, so that backup IDs greater than 2^{31} wrapped around to negative values. This issue also affected out-of-range backup IDs, which wrapped around to negative values instead of causing errors as expected in such cases. The backup ID is now treated as an unsigned value, and `ndb_mgm` now performs proper range checking for backup ID values greater than `MAX_BACKUPS` (2^{32}). (Bug #16585497, Bug #68798)

- When trying to specify a backup ID greater than the maximum allowed, the value was silently truncated. (Bug #16585455, Bug #68796)

- The unexpected shutdown of another data node as a starting data node received its node ID caused the latter to hang in Start Phase 1. (Bug #16007980)

  References: See also: Bug #18993037.

- `SELECT ... WHERE ... LIKE` from an `NDB` table could return incorrect results when using `engine_condition_pushdown=ON`. (Bug #15923467, Bug #67724)

- The `NDB` receive thread waited unnecessarily for additional job buffers to become available when receiving data. This caused the receive mutex to be held during this wait, which could result in a busy wait when the receive thread was running with real-time priority.

  This fix also handles the case where a negative return value from the initial check of the job buffer by the receive thread prevented further execution of data reception, which could possibly lead to communication blockage or configured `ReceiveBufferMemory` underutilization. (Bug #15907515)

- When the available job buffers for a given thread fell below the critical threshold, the internal multithreading job scheduler waited for job buffers for incoming rather than outgoing signals to become available, which meant that the scheduler waited the maximum timeout (1 millisecond) before resuming execution. (Bug #15907122)

- Under some circumstances, a race occurred where the wrong watchdog state could be reported. A new state name `Packing Send Buffers` is added for watchdog state number 11, previously reported as `Unknown place`. As part of this fix, the state numbers for states without names are always now reported in such cases. (Bug #14824490)

- Creating more than 32 hash maps caused data nodes to fail. Usually new hashmaps are created only when performing reorganzation after data nodes have been added or when explicit partitioning is used, such as when creating a table with the `MAX_ROWS` option, or using `PARTITION BY KEY() PARTITIONS n`. (Bug #14710311)

- When a node fails, the Distribution Handler (`DBDIH` kernel block) takes steps together with the Transaction Coordinator (`DBTC`) to make sure that all ongoing transactions involving the failed node are taken over by a surviving node and either committed or aborted. Transactions taken over which are then committed belong in the epoch that is current at the time the node failure occurs, so the surviving nodes must keep this epoch available until the transaction takeover is complete. This is needed to maintain ordering between epochs.

  A problem was encountered in the mechanism intended to keep the current epoch open which led to a race condition between this mechanism and that normally used to declare the end of an epoch. This could cause the current epoch to be closed prematurely, leading to failure of one or more surviving data nodes. (Bug #14623333, Bug #16990394)

- When performing an `INSERT ... ON DUPLICATE KEY UPDATE` on an `NDB` table where the row to be inserted already existed and was locked by another transaction, the error message returned from the `INSERT` following the timeout was `Transaction already aborted` instead of the expected `Lock wait timeout exceeded`. (Bug #14065831, Bug #65130)

- When using dynamic listening ports for accepting connections from API nodes, the port numbers were reported to the management server serially. This required a round trip for each API node, causing the time required for data nodes to connect to the management server to grow linearly with the number of API nodes. To correct this problem, each data node now reports all dynamic ports at once. (Bug #12593774)

- `ndb_error-reporter` did not support the `--help` option. (Bug #11756666, Bug #48606)

  References: See also: Bug #11752792, Bug #44082.

- When `START BACKUP WAIT STARTED` was run from the command line using `ndb_mgm --execute` (`-e`), the client did not exit until the backup completed. (Bug #11752837, Bug #44146)

- Formerly, the node used as the coordinator or leader for distributed decision making between nodes (also known as the `DICT` manager—see The DBDICT Block) was indicated in the output of the `ndb_mgm` client `SHOW` command as the “master” node, although this node has no relationship to a master server in MySQL Replication. (It should also be noted that it is not necessary to know which node is the leader except when debugging `NDBCLUSTER` source code.) To avoid possible confusion, this label has been removed, and the leader node is now indicated in `SHOW` command output using an asterisk (`*`) character. (Bug #11746263, Bug #24880)

- Program execution failed to break out of a loop after meeting a desired condition in a number of internal methods, performing unneeded work in all cases where this occurred. (Bug #69610, Bug #69611, Bug #69736, Bug #17030606, Bug #17030614, Bug #17160263)

- `ABORT BACKUP` in the `ndb_mgm` client (see Commands in the NDB Cluster Management Client) took an excessive amount of time to return (approximately as long as the backup would have required to complete, had it not been aborted), and failed to remove the files that had been generated by the aborted backup. (Bug #68853, Bug #17719439)

- Attribute promotion and demotion when restoring data to NDB tables using ndb_restore --restore-data with the --promote-attributes and --lossy-conversions options has been improved as follows:

  - Columns of types CHAR, and VARCHAR can now be promoted to BINARY and VARBINARY, and columns of the latter two types can be demoted to one of the first two.

    Note that converted character data is not checked to conform to any character set.

  - Any of the types CHAR, VARCHAR, BINARY, and VARBINARY can now be promoted to TEXT or BLOB.

    When performing such promotions, the only other sort of type conversion that can be performed at the same time is between character types and binary types.

Changes in MySQL NDB Cluster 7.2.13 (5.5.31-ndb-7.2.13) (2013-06-14, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **NDB Cluster APIs:** Added DUMP code 2514, which provides information about counts of transaction objects per API node. For more information, see DUMP 2514. See also Commands in the NDB Cluster Management Client. (Bug #15878085)

- When ndb_restore fails to find a table, it now includes in the error output an NDB API error code giving the reason for the failure. (Bug #16329067)

Bugs Fixed

- **Incompatible Change; NDB Replication:** The default value for the binlog_format system variable when the NDB storage engine is enabled is now MIXED. If NDB is not enabled, the default binary logging format is STATEMENT, as in the standard MySQL Server. (Bug #16417224)

- **NDB Disk Data:** NDB error 899 RowId already allocated was raised due to a RowId “leak” which occurred under either of the following sets of circumstances:

  1. Insertion of a row into an in-memory table was rejected after an ordered index update failed due to insufficient DataMemory.

  2. Insertion of a row into a Disk Data table was rejected due to lack of sufficient table space.

  (Bug #13927679)

  References: See also: Bug #22494024, Bug #13990924.

- The NDB Error-Reporting Utility (ndb_error_reporter) failed to include the cluster nodes' log files in the archive it produced when the FILE option was set for the parameter LogDestination. (Bug #16765651)

  References: See also: Bug #11752792, Bug #44082.

- A WHERE condition that contained a boolean test of the result of an IN subselect was not evaluated correctly. (Bug #16678033)

- In some cases a data node could stop with an exit code but no error message other than `(null)` was logged. (This could occur when using `ndbd` or `ndbmtd` for the data node process.) Now in such cases the appropriate error message is used instead (see Data Node Error Messages). (Bug #16614114)
- `mysql_upgrade` failed when run on an SQL node where distributed privileges were in use. (Bug #16226274)
- Improved handling of lagging row change event subscribers by setting size of the GCP pool to the value of `MaxBufferedEpochs`. This fix also introduces a new `MaxBufferedEpochBytes` data node configuration parameter, which makes it possible to set a total number of bytes per node to be reserved for buffering epochs. In addition, a new `DUMP` code (8013) has been added which causes a list a lagging subscribers for each node to be printed to the cluster log (see DUMP 8013). (Bug #16203623)
- Purging the binary logs could sometimes cause `mysqld` to crash. (Bug #15854719)
- An error message in `src/mgmsrv/MgmtSrvr.cpp` was corrected. (Bug #14548052, Bug #66518)
- The help text for `ndb_select_count` did not include any information about using table names. (Bug #11755737, Bug #47551)

Changes in MySQL NDB Cluster 7.2.12 (5.5.30-ndb-7.2.12) (2013-03-27, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- The length of time a management node waits for a heartbeat message from another management node is now configurable using the `HeartbeatIntervalMgmdMgmd` management node configuration parameter added in this release. The connection is considered dead after 3 missed heartbeats. The default value is 1500 milliseconds, or a timeout of approximately 6000 ms. (Bug #17807768, Bug #16426805)

Bugs Fixed

- **NDB Cluster APIs:** Trying to access old rows while using `role=ndb-caching` after restarting the memached daemon caused the daemon to crash.

  The current fix prevents the crash in such cases but does not provide full support for `role=ndb-caching`. (Bug #16204844)

Changes in MySQL NDB Cluster 7.2.11 (5.5.29-ndb-7.2.11) (Not released)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Following an upgrade to MySQL NDB Cluster 7.2.7 or later, it was not possible to downgrade online again to any previous version, due to a change in that version in the default size (number of LDM threads used) for `NDB` table hash maps. The fix for this issue makes the size configurable, with the addition of the `DefaultHashMapSize` configuration parameter.

  To retain compatibility with an older release that does not support large hash maps, you can set this parameter in the cluster' `config.ini` file to the value used in older releases (240) before performing an

upgrade, so that the data nodes continue to use smaller hash maps that are compatible with the older release. You can also now employ this parameter in MySQL NDB Cluster 7.0 and MySQL NDB Cluster 7.1 to enable larger hash maps prior to upgrading to MySQL NDB Cluster 7.2. For more information, see the description of the `DefaultHashMapSize` parameter. (Bug #14800539)

References: See also: Bug #14645319.

Bugs Fixed

- **Important Change; NDB Cluster APIs:** When checking—as part of evaluating an `if` predicate—which error codes should be propagated to the application, any error code less than 6000 caused the current row to be skipped, even those codes that should have caused the query to be aborted. In addition, a scan that aborted due to an error from `DBTUP` when no rows had been sent to the API caused `DBLQH` to send a `SCAN_FRAGCONF` signal rather than a `SCAN_FRAGREF` signal to `DBTC`. This caused `DBTC` to time out waiting for a `SCAN_FRAGREF` signal that was never sent, and the scan was never closed.

  As part of this fix, the default `ErrorCode` value used by `NdbInterpretedCode::interpret_exit_nok()` has been changed from 899 (`Rowid already allocated`) to 626 (`Tuple did not exist`). The old value continues to be supported for backward compatibility. User-defined values in the range 6000-6999 (inclusive) are also now supported. You should also keep in mind that the result of using any other `ErrorCode` value not mentioned here is not defined or guaranteed.

  See also The NDB Communication Protocol, and NDB Kernel Blocks, for more information. (Bug #16176006)

- **NDB Cluster APIs:** The `Ndb::computeHash()` API method performs a `malloc()` if no buffer is provided for it to use. However, it was assumed that the memory thus returned would always be suitably aligned, which is not always the case. Now when `malloc()` provides a buffer to this method, the buffer is aligned after it is allocated, and before it is used. (Bug #16484617)

- When using tables having more than 64 fragments in a MySQL NDB Cluster where multiple TC threads were configured (on data nodes running `ndbmtd`, using `ThreadConfig`), `AttrInfo` and `KeyInfo` memory could be freed prematurely, before scans relying on these objects could be completed, leading to a crash of the data node. (Bug #16402744)

  References: See also: Bug #13799800. This issue is a regression of: Bug #14143553.

- When started with `--initial` and an invalid `--config-file` (`-f`) option, `ndb_mgmd` removed the old configuration cache before verifying the configuration file. Now in such cases, `ndb_mgmd` first checks for the file, and continues with removing the configuration cache only if the configuration file is found and is valid. (Bug #16299289)

- Executing a `DUMP 2304` command during a data node restart could cause the data node to crash with a `Pointer too large` error. (Bug #16284258)

- Including a table as a part of a pushed join should be rejected if there are outer joined tables in between the table to be included and the tables with which it is joined with; however the check as performed for any such outer joined tables did so by checking the join type against the root of the pushed query, rather than the common ancestor of the tables being joined. (Bug #16199028)

  References: See also: Bug #16198866.

- Some queries were handled differently with `ndb_join_pushdown` enabled, due to the fact that outer join conditions were not always pruned correctly from joins before they were pushed down. (Bug #16198866)

References: See also: Bug #16199028.

- Data nodes could fail during a system restart when the host ran short of memory, due to signals of the wrong types (`ROUTE_ORD` and `TRANSID_AI_R`) being sent to the `DBSPJ` kernel block. (Bug #16187976)

- Attempting to perform additional operations such as `ADD COLUMN` as part of an `ALTER [ONLINE | OFFLINE] TABLE ... RENAME ...` statement is not supported, and now fails with an **`ER_NOT_SUPPORTED_YET`** error. (Bug #16021021)

- Due to a known issue in the MySQL Server, it is possible to drop the `PERFORMANCE_SCHEMA` database. (Bug #15831748) In addition, when executed on a MySQL Server acting as a MySQL NDB Cluster SQL node, `DROP DATABASE` caused this database to be dropped on all SQL nodes in the cluster. Now, when executing a distributed drop of a database, `NDB` does not delete tables that are local only. This prevents MySQL system databases from being dropped in such cases. (Bug #14798043)

  References: See also: Bug #15831748.

- When performing large numbers of DDL statements (100 or more) in succession, adding an index to a table sometimes caused `mysqld` to crash when it could not find the table in `NDB`. Now when this problem occurs, the DDL statement should fail with an appropriate error.

  A workaround in such cases may be to create the table with the index as part of the initial `CREATE TABLE`, rather than adding the index in a subsequent `ALTER TABLE` statement. (Bug #14773491)

- Executing `OPTIMIZE TABLE` on an `NDB` table containing `TEXT` or `BLOB` columns could sometimes cause `mysqld` to fail. (Bug #14725833)

- Executing a `DUMP 1000` command that contained extra or malformed arguments could lead to data node failures. (Bug #14537622)

- Exhaustion of `LongMessageBuffer` memory under heavy load could cause data nodes running `ndbmtd` to fail. (Bug #14488185)

- The `ndb_mgm` client `HELP` command did not show the complete syntax for the `REPORT` command.

Changes in MySQL NDB Cluster 7.2.10 (5.5.29-ndb-7.2.10) (2013-01-02, General Availability)

Functionality Added or Changed

- Added several new columns to the `transporters` table and counters for the `counters` table of the `ndbinfo` information database. The information provided may help in troublehsooting of transport overloads and problems with send buffer memory allocation. For more information, see the descriptions of these tables. (Bug #15935206)

- To provide information which can help in assessing the current state of arbitration in a MySQL NDB Cluster as well as in diagnosing and correcting arbitration problems, 3 new tables—`membership`, `arbitrator_validity_detail`, and `arbitrator_validity_summary`—have been added to the `ndbinfo` information database. (Bug #13336549)

Bugs Fixed

- When an NDB table grew to contain approximately one million rows or more per partition, it became possible to insert rows having duplicate primary or unique keys into it. In addition, primary key lookups began to fail, even when matching rows could be found in the table by other means.

  This issue was introduced in MySQL NDB Cluster 7.0.36, MySQL NDB Cluster 7.1.26, and MySQL NDB Cluster 7.2.9. Signs that you may have been affected include the following:

  - Rows left over that should have been deleted
  - Rows unchanged that should have been updated
  - Rows with duplicate unique keys due to inserts or updates (which should have been rejected) that failed to find an existing row and thus (wrongly) inserted a new one

  This issue does not affect simple scans, so you can see all rows in a given *`table`* using `SELECT * FROM` *`table`* and similar queries that do not depend on a primary or unique key.

  Upgrading to or downgrading from an affected release can be troublesome if there are rows with duplicate primary or unique keys in the table; such rows should be merged, but the best means of doing so is application dependent.

  In addition, since the key operations themselves are faulty, a merge can be difficult to achieve without taking the MySQL NDB Cluster offline, and it may be necessary to dump, purge, process, and reload the data. Depending on the circumstances, you may want or need to process the dump with an external application, or merely to reload the dump while ignoring duplicates if the result is acceptable.

  Another possibility is to copy the data into another table without the original table' unique key constraints or primary key (recall that `CREATE TABLE t2 SELECT * FROM t1` does not by default copy `t1`'s primary or unique key definitions to `t2`). Following this, you can remove the duplicates from the copy, then add back the unique constraints and primary key definitions. Once the copy is in the desired state, you can either drop the original table and rename the copy, or make a new dump (which can be loaded later) from the copy. (Bug #16023068, Bug #67928)

- The management client command `ALL REPORT BackupStatus` failed with an error when used with data nodes having multiple LQH worker threads (`ndbmtd` data nodes). The issue did not effect the *`node_id`* `REPORT BackupStatus` form of this command. (Bug #15908907)

- The multithreaded job scheduler could be suspended prematurely when there were insufficient free job buffers to allow the threads to continue. The general rule in the job thread is that any queued messages should be sent before the thread is allowed to suspend itself, which guarantees that no other threads or API clients are kept waiting for operations which have already completed. However, the number of messages in the queue was specified incorrectly, leading to increased latency in delivering signals, sluggish response, or otherwise suboptimal performance. (Bug #15908684)

- The setting for the `DefaultOperationRedoProblemAction` API node configuration parameter was ignored, and the default value used instead. (Bug #15855588)

- Node failure during the dropping of a table could lead to the node hanging when attempting to restart.

  When this happened, the NDB internal dictionary (`DBDICT`) lock taken by the drop table operation was held indefinitely, and the logical global schema lock taken by the SQL the drop table operation from which the drop operation originated was held until the NDB internal operation timed out. To aid in debugging such occurrences, a new dump code, `DUMP 1228` (or `DUMP DictDumpLockQueue`), which dumps the contents of the `DICT` lock queue, has been added in the `ndb_mgm` client. (Bug #14787522)

- Job buffers act as the internal queues for work requests (signals) between block threads in `ndbmtd` and could be exhausted if too many signals are sent to a block thread.

  Performing pushed joins in the `DBSPJ` kernel block can execute multiple branches of the query tree in parallel, which means that the number of signals being sent can increase as more branches are executed. If `DBSPJ` execution cannot be completed before the job buffers are filled, the data node can fail.

  This problem could be identified by multiple instances of the message `sleeploop 10!!` in the cluster out log, possibly followed by `job buffer full`. If the job buffers overflowed more gradually, there could also be failures due to error 1205 (`Lock wait timeout exceeded`), shutdowns initiated by the watchdog timer, or other timeout related errors. These were due to the slowdown caused by the 'sleeploop'.

  Normally up to a 1:4 fanout ratio between consumed and produced signals is permitted. However, since there can be a potentially unlimited number of rows returned from the scan (and multiple scans of this type executing in parallel), any ratio greater 1:1 in such cases makes it possible to overflow the job buffers.

  The fix for this issue defers any lookup child which otherwise would have been executed in parallel with another is deferred, to resume when its parallel child completes one of its own requests. This restricts the fanout ratio for bushy scan-lookup joins to 1:1. (Bug #14709490)

  References: See also: Bug #14648712.

- During an online upgrade, certain SQL statements could cause the server to hang, resulting in the error `Got error 4012 'Request ndbd time-out, maybe due to high load or communication problems' from NDBCLUSTER`. (Bug #14702377)

- The recently added LCP fragment scan watchdog occasionally reported problems with LCP fragment scans having very high table id, fragment id, and row count values.

  This was due to the watchdog not accounting for the time spent draining the backup buffer used to buffer rows before writing to the fragment checkpoint file.

  Now, in the final stage of an LCP fragment scan, the watchdog switches from monitoring rows scanned to monitoring the buffer size in bytes. The buffer size should decrease as data is written to the file, after which the file should be promptly closed. (Bug #14680057)

- Under certain rare circumstances, MySQL NDB Cluster data nodes could crash in conjunction with a configuration change on the data nodes from a single-threaded to a multithreaded transaction coordinator (using the `ThreadConfig` configuration parameter for `ndbmtd`). The problem occurred when a `mysqld` that had been started prior to the change was shut down following the rolling restart of the data nodes required to effect the configuration change. (Bug #14609774)

Changes in MySQL NDB Cluster 7.2.9 (5.5.28-ndb-7.2.9) (2012-11-22, General Availability)

Functionality Added or Changed

- **Important Change; MySQL NDB ClusterJ:** A new `CMake` option `WITH_NDB_JAVA` is introduced. When this option is enabled, the MySQL NDB Cluster build is configured to include Java support, including

support for `ClusterJ`. If the JDK cannot be found, `CMake` fails with an error. This option is enabled by default; if you do *not* wish to compile Java support into MySQL NDB Cluster, you must now set this explicitly when configuring the build, using `-DWITH_NDB_JAVA=OFF`. (Bug #12379755)

- Added 3 new columns to the `transporters` table in the `ndbinfo` database. The `remote_address`, `bytes_sent`, and `bytes_received` columns help to provide an overview of data transfer across the transporter links in a MySQL NDB Cluster. This information can be useful in verifying system balance, partitioning, and front-end server load balancing; it may also be of help when diagnosing network problems arising from link saturation, hardware faults, or other causes. (Bug #14685458)

- Data node logs now provide tracking information about arbitrations, including which nodes have assumed the arbitrator role and at what times. (Bug #11761263, Bug #53736)

Bugs Fixed

- **NDB Disk Data:** Concurrent DML and DDL operations against the same NDB table could cause `mysqld` to crash. (Bug #14577463)

- A slow filesystem during local checkpointing could exert undue pressure on `DBDIH` kernel block file page buffers, which in turn could lead to a data node crash when these were exhausted. This fix limits the number of table definition updates that `DBDIH` can issue concurrently. (Bug #14828998)

- The management server process, when started with `--config-cache=FALSE`, could sometimes hang during shutdown. (Bug #14730537)

- The output from `ndb_config --configinfo` now contains the same information as that from `ndb_config --configinfo --xml`, including explicit indicators for parameters that do not require restarting a data node with `--initial` to take effect. In addition, `ndb_config` indicated incorrectly that the `LogLevelCheckpoint` data node configuration parameter requires an initial node restart to take effect, when in fact it does not; this error was also present in the MySQL NDB Cluster documentation, where it has also been corrected. (Bug #14671934)

- Attempting to restart more than 5 data nodes simultaneously could cause the cluster to crash. (Bug #14647210)

- In MySQL NDB Cluster 7.2.7, the size of the hash map was increased to 3840 LDM threads. However, when upgrading a MySQL NDB Cluster from a previous release, existing tables could not use or be modified online to take advantage of the new size, even when the number of fragments was increased by (for example) adding new data nodes to the cluster. Now in such cases, following an upgrade and (once the number of fragments has been increased), you can run `ALTER TABLE ... REORGANIZE PARTITION` on tables that were created in MySQL NDB Cluster 7.2.6 or earlier, after which they can use the larger hash map size. (Bug #14645319)

- Concurrent `ALTER TABLE` with other DML statements on the same NDB table returned `Got error -1 'Unknown error code' from NDBCLUSTER`. (Bug #14578595)

- CPU consumption peaked several seconds after the forced termination an NDB client application due to the fact that the DBTC kernel block waited for any open transactions owned by the disconnected API client to be terminated in a busy loop, and did not break between checks for the correct state. (Bug #14550056)

- Receiver threads could wait unnecessarily to process incomplete signals, greatly reducing performance of `ndbmtd`. (Bug #14525521)

- On platforms where epoll was not available, setting multiple receiver threads with the `ThreadConfig` parameter caused `ndbmtd` to fail. (Bug #14524939)

- Setting `BackupMaxWriteSize` to a very large value as compared with `DiskCheckpointSpeed` caused excessive writes to disk and CPU usage. (Bug #14472648)

- Added the `--connect-retries` and `--connect-delay` startup options for `ndbd` and `ndbmtd`. `--connect-retries` (default 12) controls how many times the data node tries to connect to a management server before giving up; setting it to -1 means that the data node never stops trying to make contact. `--connect-delay` sets the number of seconds to wait between retries; the default is 5. (Bug #14329309, Bug #66550)

- Following a failed `ALTER TABLE ... REORGANIZE PARTITION` statement, a subsequent execution of this statement after adding new data nodes caused a failure in the `DBDIH` kernel block which led to an unplanned shutdown of the cluster.

  `DUMP` code 7019 was added as part of this fix. It can be used to obtain diagnostic information relating to a failed data node. See DUMP 7019, for more information. (Bug #14220269)

  References: See also: Bug #18550318.

- It was possible in some cases for two transactions to try to drop tables at the same time. If the master node failed while one of these operations was still pending, this could lead either to additional node failures (and cluster shutdown) or to new dictionary operations being blocked. This issue is addressed by ensuring that the master will reject requests to start or stop a transaction while there are outstanding dictionary takeover requests. In addition, table-drop operations now correctly signal when complete, as the `DBDICT` kernel block could not confirm node takeovers while such operations were still marked as pending completion. (Bug #14190114)

- The `DBSPJ` kernel block had no information about which tables or indexes actually existed, or which had been modified or dropped, since execution of a given query began. Thus, `DBSPJ` might submit dictionary requests for nonexistent tables or versions of tables, which could cause a crash in the `DBDIH` kernel block.

  This fix introduces a simplified dictionary into the `DBSPJ` kernel block such that `DBSPJ` can now check reliably for the existence of a particular table or version of a table on which it is about to request an operation. (Bug #14103195)

- Previously, it was possible to store a maximum of 46137488 rows in a single MySQL NDB Cluster partition. This limitation has now been removed. (Bug #13844405, Bug #14000373)

  References: See also: Bug #13436216.

- When using `ndbmtd` and performing joins, data nodes could fail where `ndbmtd` processes were configured to use a large number of local query handler threads (as set by the `ThreadConfig` configuration parameter), the tables accessed by the join had a large number of partitions, or both. (Bug #13799800, Bug #14143553)

Changes in MySQL NDB Cluster 7.2.8 (5.5.27-ndb-7.2.8) (2012-09-07, General Availability)

Bugs Fixed

- An improvement introduced in MySQL NDB Cluster 7.2.7 saves memory used by buffers for communications between threads. One way in which this was implemented was by not allocating buffers between threads which were assumed not to communicate, including communication between local query handler (LQH or LDM) threads.

However, the BACKUP kernel block is used by the LDM thread, and during NDB native backup the first instance of this block used by the first LDM thread acts as a client coordinator, and thus attempts to communicate with other LDM threads. This caused the START BACKUP command to fail when using ndbmtd configured with multiple LDM threads.

The fix for this issue restores the buffer used for communication between LDM threads in such cases. (Bug #14489398)

- When reloading the redo log during a node or system restart, and with NoOfFragmentLogFiles greater than or equal to 42, it was possible for metadata to be read for the wrong file (or files). Thus, the node or nodes involved could try to reload the wrong set of data. (Bug #14389746)
- When an NDB table was created during a data node restart, the operation was rolled back in the NDB engine, but not on the SQL node where it was executed. This was due to the table .FRM files not being cleaned up following the operation that was rolled back by NDB. Now in such cases these files are removed. (Bug #13824846)

Changes in MySQL NDB Cluster 7.2.7 (5.5.25a-ndb-7.2.7) (2012-07-17, General Availability)

Bugs Fixed

- **Important Change:** When FILE was used for the value of the LogDestination parameter without also specifying the filename, the log file name defaulted to logger.log. Now in such cases, the name defaults to ndb_*nodeid*_cluster.log. (Bug #11764570, Bug #57417)
- **Packaging; Solaris:** Some builds on Solaris 64-bit failed because the packages exceeded the 2GB limit for the SVR4 installation layout. Now such packages are built without the embedded versions of mysqltest and mysql-client_test to save space. (Bug #14058643)
- **Packaging; NDB Cluster APIs:** The file META-INF/services/org.apache.openjpa.lib.conf.ProductDerivation was missing from the clusterjpa JAR file. This could cause setting openjpa.BrokerFactory to “ndb” to be rejected. (Bug #14192154)

  References: See also: Bug #52106.
- **NDB Cluster APIs:** libndbclient did not include the NdbScanFilter class. (Bug #14010507)
- **NDB Cluster APIs:** When an NDB API application called NdbScanOperation::nextResult() again after the previous call had returned end-of-file (return code 1), a transaction object was leaked. Now when this happens, NDB returns error code 4210 (Ndb sent more info than length specified); previouslyu in such cases, -1 was returned. In addition, the extra transaction object associated with the scan is freed, by returning it to the transaction coordinator's idle list. (Bug #11748194)
- If the Transaction Coordinator aborted a transaction in the “prepared” state, this could cause a resource leak. (Bug #14208924)
- When attempting to connect using a socket with a timeout, it was possible (if the timeout was exceeded) for the socket not to be set back to blocking. (Bug #14107173)
- A shortage of scan fragment records in DBTC resulted in a leak of concurrent scan table records and key operation records. (Bug #13966723)
- Attempting to add both a column and an index on that column in the same online ALTER TABLE statement caused mysqld to fail. Although this issue affected only the mysqld shipped with MySQL

NDB Cluster, the table named in the `ALTER TABLE` could use any storage engine for which online operations are supported. (Bug #12755722)

Changes in MySQL NDB Cluster 7.2.6 (5.5.22-ndb-7.2.6) (2012-05-21, General Availability)

Bugs Fixed

- **Important Change:** The `ALTER ONLINE TABLE ... REORGANIZE PARTITION` statement can be used to create new table partitions after new empty nodes have been added to a MySQL NDB Cluster. Usually, the number of partitions to create is determined automatically, such that, if no new partitions are required, then none are created. This behavior can be overridden by creating the original table using the `MAX_ROWS` option, which indicates that extra partitions should be created to store a large number of rows. However, in this case `ALTER ONLINE TABLE ... REORGANIZE PARTITION` simply uses the `MAX_ROWS` value specified in the original `CREATE TABLE` statement to determine the number of partitions required; since this value remains constant, so does the number of partitions, and so no new ones are created. This means that the table is not rebalanced, and the new data nodes remain empty.

  To solve this problem, support is added for `ALTER ONLINE TABLE ... MAX_ROWS=`*`newvalue`*, where *`newvalue`* is greater than the value used with `MAX_ROWS` in the original `CREATE TABLE` statement. This larger `MAX_ROWS` value implies that more partitions are required; these are allocated on the new data nodes, which restores the balanced distribution of the table data.

  For more information, see ALTER TABLE Statement, and Adding NDB Cluster Data Nodes Online. (Bug #13714648)

- **NDB Cluster APIs:** An assert in `memcache/include/Queue.h` by `NDB` could cause `memcached` to fail. (Bug #13874027)

- An error handling routine in the local query handler (`DBLQH`) used the wrong code path, which could corrupt the transaction ID hash, causing the data node process to fail. This could in some cases possibly lead to failures of other data nodes in the same node group when the failed node attempted to restart. (Bug #14083116)

- When a fragment scan occurring as part of a local checkpoint (LCP) stopped progressing, this kept the entire LCP from completing, which could result it redo log exhaustion, write service outage, inability to recover nodes, and longer system recovery times. To help keep this from occurring, MySQL NDB Cluster now implements an LCP watchdog mechanism, which monitors the fragment scans making up the LCP and takes action if the LCP is observed to be delinquent.

  This is intended to guard against any scan related system-level I/O errors or other issues causing problems with LCP and thus having a negative impact on write service and recovery times. Each node independently monitors the progress of local fragment scans occurring as part of an LCP. If no progress is made for 20 seconds, warning logs are generated every 10 seconds thereafter for up to 1 minute. At this point, if no progress has been made, the fragment scan is considered to have hung, and the node is restarted to enable the LCP to continue.

  In addition, a new `ndbd` exit code `NDBD_EXIT_LCP_SCAN_WATCHDOG_FAIL` is added to identify when this occurs. See LQH Errors, for more information. (Bug #14075825)

- It could sometimes happen that a query pushed down to the data nodes could refer to buffered rows which had been released, and possibly overwritten by other rows. Such rows, if overwritten, could lead to incorrect results from a pushed query, and possibly even to failure of one or more data nodes or SQL nodes. (Bug #14010406)

- `DUMP 2303` in the `ndb_mgm` client now includes the status of the single fragment scan record reserved for a local checkpoint. (Bug #13986128)
- Pushed joins performed as part of a stored procedure or trigger could cause spurious `Out of memory` errors on the SQL node where they were executed. (Bug #13945264)

  References: See also: Bug #13944272.
- `INSERT ... SELECT` executed inside a trigger or stored procedure with `ndb_join_pushdown` enabled could lead to a crash of the SQL node on which it was run. (Bug #13901890)

  References: See also: Bug #13945264.
- When upgrading or downgrading between a MySQL NDB Cluster version supporting distributed pushdown joins (MySQL NDB Cluster 7.2 and later) and one that did not, queries that the later MySQL NDB Cluster version tried to push down could cause data nodes still running the earlier version to fail. Now the SQL nodes check the version of the software running on the data nodes, so that queries are not pushed down if there are any data nodes in the cluster that do not support pushdown joins. (Bug #13894817)
- `ndbmemcached` exited unexpectedly when more than 128 clients attempted to connect concurrently using prefixes. In addition, a `NOT FOUND` error was returned when the `memcached` engine encountered a temporary error from `NDB`; now the error `No Ndb Instances in freelist` is returned instead. (Bug #13890064, Bug #13891085)
- The performance of `ndbmemcache` with a workload that consisted mostly of primary key reads became degraded. (Bug #13868787, Bug #64713)
- When the `--skip-config-cache` and `--initial` options were used together, `ndb_mgmd` failed to start. (Bug #13857301)
- The memcached server failed to build correctly on 64-bit Solaris/SPARC. (Bug #13854122)
- `ALTER ONLINE TABLE` failed when a `DEFAULT` option was used. (Bug #13830980)
- In some cases, restarting data nodes spent a very long time in Start Phase 101, when API nodes must connect to the starting node (using `NdbEventOperation`), when the API nodes trying to connect failed in a live-lock scenario. This connection process uses a handshake during which a small number of messages are exchanged, with a timeout used to detect failures during the handshake.

  Prior to this fix, this timeout was set such that, if one API node encountered the timeout, all other nodes connecting would do the same. The fix also decreases this timeout. This issue (and the effects of the fix) are most likely to be observed on relatively large configurations having 10 or more data nodes and 200 or more API nodes. (Bug #13825163)
- `ndbmtd` failed to restart when the size of a table definition exceeded 32K.

  (The size of a table definition is dependent upon a number of factors, but in general the 32K limit is encountered when a table has 250 to 300 columns.) (Bug #13824773)
- An initial start using `ndbmtd` could sometimes hang. This was due to a state which occurred when several threads tried to flush a socket buffer to a remote node. In such cases, to minimize flushing of socket buffers, only one thread actually performs the send, on behalf of all threads. However, it was possible in certain cases for there to be data in the socket buffer waiting to be sent with no thread ever being chosen to perform the send. (Bug #13809781)
- When trying to use `ndb_size.pl --hostname=`*`host`*`:`*`port`* to connect to a MySQL server running on a nonstandard port, the *`port`* argument was ignored. (Bug #13364905, Bug #62635)

- The `transaction_allow_batching` server system variable was inadvertently removed from the NDB 7.2 codebase prior to General Availability. This fix restores the variable. (Bug #64697, Bug #13891116, Bug #13947227)

Changes in MySQL NDB Cluster 7.2.5 (5.5.20-ndb-7.2.5) (2012-03-20, General Availability)

Bugs Fixed

- **Important Change:** A number of changes have been made in the configuration of transporter send buffers.

  1. The data node configuration parameter `ReservedSendBufferMemory` is now deprecated, and thus subject to removal in a future MySQL NDB Cluster release. `ReservedSendBufferMemory` has been non-functional since it was introduced and remains so.

  2. `TotalSendBufferMemory` now works correctly with data nodes using `ndbmtd`.

  3. `SendBufferMemory` can now over-allocate into `SharedGlobalMemory` for `ndbmtd` data nodes (only).

  4. A new data node configuration parameter `ExtraSendBufferMemory` is introduced. Its purpose is to control how much additional memory can be allocated to the send buffer over and above that specified by `TotalSendBufferMemory` or `SendBufferMemory`. The default setting (0) allows up to 16MB to be allocated automatically.

  (Bug #13633845, Bug #11760629, Bug #53053)

- Setting `insert_id` had no effect on the auto-increment counter for `NDB` tables. (Bug #13731134)

- A data node crashed when more than 16G fixed-size memory was allocated by `DBTUP` to one fragment (because the `DBACC` kernel block was not prepared to accept values greater than 32 bits from it, leading to an overflow). Now in such cases, the data node returns Error 889 `Table fragment fixed data reference has reached maximum possible value...`. When this happens, you can work around the problem by increasing the number of partitions used by the table (such as by using the `MAXROWS` option with `CREATE TABLE`). (Bug #13637411)

  References: See also: Bug #11747870, Bug #34348.

- Several instances in the NDB code affecting the operation of multithreaded data nodes, where `SendBufferMemory` was associated with a specific thread for an unnecessarily long time, have been identified and fixed, by minimizing the time that any of these buffers can be held exclusively by a given thread (send buffer memory being critical to operation of the entire node). (Bug #13618181)

- A very large value for `BackupWriteSize`, as compared to `BackupMaxWriteSize`, `BackupDataBufferSize`, or `BackupLogBufferSize`, could cause a local checkpoint or backup to hang. (Bug #13613344)

- Queries using `LIKE ... ESCAPE` on `NDB` tables failed when pushed down to the data nodes. Such queries are no longer pushed down, regardless of the value of `engine_condition_pushdown`. (Bug #13604447, Bug #61064)

- To avoid TCP transporter overload, an overload flag is kept in the NDB kernel for each data node; this flag is used to abort key requests if needed, yielding error 1218 `Send Buffers overloaded in NDB kernel` in such cases. Scans can also put significant pressure on transporters, especially where scans with a high degree of parallelism are executed in a configuration with relatively small send buffers.

However, in these cases, overload flags were not checked, which could lead to node failures due to send buffer exhaustion. Now, overload flags are checked by scans, and in cases where returning sufficient rows to match the batch size (`--ndb-batch-size` server option) would cause an overload, the number of rows is limited to what can be accommodated by the send buffer.

See also Configuring NDB Cluster Send Buffer Parameters. (Bug #13602508)

- A `SELECT` from an `NDB` table using `LIKE` with a multibyte column (such as `utf8`) did not return the correct result when `engine_condition_pushdown` was enabled. (Bug #13579318, Bug #64039)

  References: See also: Bug #13608135.

- `memcached` started with the `ndb_engine.so` plugin failed to start if the `ndbmemcache` database was missing. (Bug #13567414)

- When starting `memcached` using `ndb_engine.so` with only 2 API slots available in the cluster configuration file, `memcached` attempted to make a third connection and crashed when this failed. (Bug #13559728)

- A node failure and recovery while performing a scan on more than 32 partitions led to additional node failures during node takeover. (Bug #13528976)

- The `--skip-config-cache` option now causes `ndb_mgmd` to skip checking for the configuration directory, and thus to skip creating it in the event that it does not exist. (Bug #13428853)

Changes in MySQL NDB Cluster 7.2.4 (5.5.19-ndb-7.2.4) (2012-02-15, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change:** A `mysqld` process joining a MySQL NDB Cluster where distributed privileges are in use now automatically executes a `FLUSH PRIVILEGES` as part of the connection process, so that the cluster's distributed privileges take immediate effect on the new SQL node. (Bug #13340854)

Bugs Fixed

- **Packaging:** RPM distributions of MySQL NDB Cluster 7.1 contained a number of packages which in MySQL NDB Cluster 7.2 have been merged into the `MySQL-Cluster-server` RPM. However, the MySQL NDB Cluster 7.2 `MySQL-Cluster-server` RPM did not actually obsolete these packages, which meant that they had to be removed manually prior to performing an upgrade from a MySQL NDB Cluster 7.1 RPM installation.

  These packages include the `MySQL-Cluster-clusterj`, `MySQL-Cluster-extra`, `MySQL-Cluster-management`, `MySQL-Cluster-storage`, and `MySQL NDB Cluster-tools` RPMs.

  For more information, see Installing NDB Cluster from RPM. (Bug #13545589)

- Accessing a table having a `BLOB` column but no primary key following a restart of the SQL node failed with Error 1 (`Unknown error code`). (Bug #13563280)

- At the beginning of a local checkpoint, each data node marks its local tables with a “to be checkpointed” flag. A failure of the master node during this process could cause either the LCP to hang, or one or more data nodes to be forcibly shut down. (Bug #13436481)

- A node failure while a `ANALYZE TABLE` statement was executing resulted in a hung connection (and the user was not informed of any error that would cause this to happen). (Bug #13416603)

  References: See also: Bug #13407848.

Changes in MySQL NDB Cluster 7.2.3 (5.5.17-ndb-7.2.3) (Not released)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Added the `ThreadConfig` data node configuration parameter to enable control of multiple threads and CPUs when using `ndbmtd`, by assigning threads of one or more specified types to execute on one or more CPUs. This can provide more precise and flexible control over multiple threads than can be obtained using the `LockExecuteThreadToCPU` parameter. (Bug #11795581)

Bugs Fixed

- Added the `MinFreePct` data node configuration parameter, which specifies a percentage of data node resources to hold in reserve for restarts. The resources monitored are `DataMemory`, `IndexMemory`, and any per-table `MAX_ROWS` settings (see CREATE TABLE Statement). The default value of `MinFreePct` is 5, which means that 5% from each these resources is now set aside for restarts. (Bug #13436216)
- Issuing `TRUNCATE TABLE` on `mysql.user`, `mysql.host`, `mysql.db`, `mysql.tables_priv`, `mysql.proxies_priv`, or `mysql.procs_priv`, when these tables had been converted to MySQL NDB Cluster distributed grant tables, caused `mysqld` to crash. (Bug #13346955)
- Restarting an SQL node configured for distributed grants could sometimes result in a crash. (Bug #13340819)
- Previously, forcing simultaneously the shutdown of multiple data nodes using `SHUTDOWN -F` in the `ndb_mgm` management client could cause the entire cluster to fail. Now in such cases, any such nodes are forced to abort immediately. (Bug #12928429)

Changes in MySQL NDB Cluster 7.2.2 (5.5.16-ndb-7.2.2) (2011-12-14, Development Milestone Release)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **NDB Cluster APIs:** Added support for the Memcache API using *ndbmemcache*, a loadable storage engine for memcached version 1.6 and later, which can be used to provide a persistent MySQL NDB Cluster data store, accessed using the memcache protocol.

  The standard `memcached` caching engine is now included in the MySQL NDB Cluster distribution. Each `memcached` server, in addition to providing direct access to data stored in a MySQL NDB Cluster, is able to cache data locally and serve (some) requests from this local cache.

  The `memcached` server can also provide an interface to existing MySQL NDB Cluster tables that is strictly defined, so that an administrator can control exactly which tables and columns are referenced by particular memcache keys and values, and which operations are allowed on these keys and values.

For more information, see ndbmemcache—Memcache API for NDB Cluster.

- Added the `ndbinfo_select_all` utility.

Bugs Fixed

- **Microsoft Windows:** The `include/storage` directory, where the header files supplied for use in compiling MySQL NDB Cluster applications are normally located, was missing from MySQL NDB Cluster release packages for Windows. (Bug #12690665)
- When adding data nodes online, if the SQL nodes were not restarted before starting the new data nodes, the next query to be executed crashed the SQL node on which it was run. (Bug #13715216, Bug #62847)

  References: This issue is a regression of: Bug #13117187.

Changes in MySQL NDB Cluster 7.2.1 (5.5.15-ndb-7.2.1) (2011-10-03, Development Milestone Release)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change:** By default, data nodes now exhibit fail-fast behavior whenever they encounter corrupted tuples—in other words, a data node forcibly shuts down whenever it detects a corrupted tuple. To override this behavior, you can disable the `CrashOnCorruptedTuple` data node configuration parameter, which is enabled by default.

  This is a change from MySQL NDB Cluster 7.0 and MySQL NDB Cluster 7.1, where `CrashOnCorruptedTuple` was disabled (so that data nodes ignored tuple corruption) by default. (Bug #12598636)
- **Important Change:** The default values for a number of MySQL NDB Cluster data node configuration parameters have changed, to provide more resiliency to environmental issues and better handling of some potential failure scenarios, and to perform more reliably with increases in memory and other resource requirements brought about by recent improvements in join handling by `NDB`. The affected parameters are listed here:
  - `HeartbeatIntervalDbDb`: Default increased from 1500 ms to 5000 ms.
  - `ArbitrationTimeout`: Default increased from 3000 ms to 7500 ms.
  - `TimeBetweenEpochsTimeout`: Now effectively disabled by default (default changed from 4000 ms to 0).
  - `SharedGlobalMemory`: Default increased from 20MB to 128MB.
  - `MaxParallelScansPerFragment`: Default increased from 32 to 256.

  In addition, when `MaxNoOfLocalScans` is not specified, the value computed for it automatically has been increased by a factor of 4 (that is, to 4 times `MaxNoOfConcurrentScans`, times the number of data nodes in the cluster).

- **Important Change:** MySQL user privileges can now be distributed automatically across all MySQL servers (SQL nodes) connected to the same MySQL NDB Cluster. Previously, each MySQL Server's user privilege tables were required to use the `MyISAM` storage engine, which meant that a user account and its associated privileges created on one SQL node were not available on any other SQL node without manual intervention. MySQL NDB Cluster now provides an SQL script file `ndb_dist_priv.sql` that can be found in `share/mysql` under the MySQL installation directory; loading this script creates a stored procedure `mysql_cluster_move_privileges` that can be used following initial installation to convert the privilege tables to the `NDB` storage engine, so that any time a MySQL user account is created, dropped, or has its privileges updated on any SQL node, the changes take effect immediately on all other MySQL servers attached to the cluster. Note that you may need to execute `FLUSH PRIVILEGES` on any SQL nodes with connected MySQL clients (or to disconnect and then reconnect the clients) in order for those clients to be able to see the changes in privileges.

  Once `mysql_cluster_move_privileges` has been executed successfully, all MySQL user privileges are distributed across all connected MySQL Servers. MySQL Servers that join the cluster after this automatically participate in the privilege distribution.

  `ndb_dist_priv.sql` also provides stored routines that can be used to verify that the privilege tables have been distributed successfully, and to perform other tasks.

  For more information, see Distributed Privileges Using Shared Grant Tables.

- **Performance:** Added support for pushing down joins to the NDB kernel for parallel execution across the data nodes, which can speed up execution of joins across `NDB` tables by a factor of 20 or more in some cases. Some restrictions apply on the types of joins that can be optimized in this way; in particular, columns to be joined must use exactly the same data type, and cannot be any of the `BLOB` or `TEXT` types. In addition, columns to be joined must be part of a table index or primary key. Support for this feature can be enabled or disabled using the `ndb_join_pushdown` server system variable (enabled by default); see the description of this variable for more information and examples.

  As part of this improvement, the status variables `Ndb_pushed_queries_defined`, `Ndb_pushed_queries_dropped`, `Ndb_pushed_queries_executed`, and `Ndb_pushed_reads` also been introduced for monitoring purposes. You can also see whether a given join is pushed down using `EXPLAIN`. In addition, several new counters relating to push-down join performance have been added to the `counters` table in the `ndbinfo` database. For more information, see the descriptions of the status variables previously mentioned.

- It is now possible to filter the output from `ndb_config` so that it displays only system, data node, or connection parameters and values, using one of the options `--system`, `--nodes`, or `--connections`, respectively. In addition, it is now possible to specify from which data node the configuration data is obtained, using the `--config_from_node` option that is added in this release.

  For more information, see `ndb_config` — Extract NDB Cluster Configuration Information. (Bug #11766870)

Bugs Fixed

- **Incompatible Change; NDB Cluster APIs:** Restarting a machine hosting data nodes, SQL nodes, or both, caused such nodes when restarting to time out while trying to obtain node IDs.

  As part of the fix for this issue, the behavior and default values for the NDB API `Ndb_cluster_connection::connect()` method have been improved. Due to these changes, the version number for the included NDB client library (`libndbclient.so`) has been increased from 4.0.0 to 5.0.0. For NDB API applications, this means that as part of any upgrade, you must do both of the following:

- Review and possibly modify any NDB API code that uses the `connect()` method, in order to take into account its changed default retry handling.
- Recompile any NDB API applications using the new version of the client library.

Also in connection with this issue, the default value for each of the two `mysqld` options `--ndb-wait-connected` and `--ndb-wait-setup` has been increased to 30 seconds (from 0 and 15, respectively). In addition, a hard-coded 30-second delay was removed, so that the value of `--ndb-wait-connected` is now handled correctly in all cases. (Bug #12543299)

- `AUTO_INCREMENT` values were not set correctly for `INSERT IGNORE` statements affecting `NDB` tables. This could lead such statements to fail with `Got error 4350 'Transaction already aborted' from NDBCLUSTER` when inserting multiple rows containing duplicate values. (Bug #11755237, Bug #46985)

Changes in MySQL NDB Cluster 7.2.0 (5.1.51-ndb-7.2.0) (2011-04-11, Development Milestone Release)

Functionality Added or Changed

- **Important Change:** The default values for a number of MySQL NDB Cluster data node configuration parameters have changed, to provide more resiliency to environmental issues and better handling of some potential failure scenarios, and to perform more reliably with increases in memory and other resource requirements brought about by recent improvements in join handling by `NDB`. The affected parameters are listed here:
  - `HeartbeatIntervalDbDb`: Default increased from 1500 ms to 5000 ms.
  - `ArbitrationTimeout`: Default increased from 3000 ms to 7500 ms.
  - `TimeBetweenEpochsTimeout`: Now effectively disabled by default (default changed from 4000 ms to 0).
  - `SharedGlobalMemory`: Default increased from 20MB to 128MB.
  - `MaxParallelScansPerFragment`: Default increased from 32 to 256.

  In addition, when `MaxNoOfLocalScans` is not specified, the value computed for it automatically has been increased by a factor of 4 (that is, to 4 times `MaxNoOfConcurrentScans`, times the number of data nodes in the cluster).

- **Important Change:** MySQL user privileges can now be distributed automatically across all MySQL servers (SQL nodes) connected to the same MySQL NDB Cluster. Previously, each MySQL Server's user privilege tables were required to use the `MyISAM` storage engine, which meant that a user account and its associated privileges created on one SQL node were not available on any other SQL node without manual intervention. MySQL NDB Cluster now provides an SQL script file `ndb_dist_priv.sql` that can be found in `share/mysql` under the MySQL installation directory; loading this script creates a stored procedure `mysql_cluster_move_privileges` that can be used following initial installation to convert the privilege tables to the `NDB` storage engine, so that any time a MySQL user account is created, dropped, or has its privileges updated on any SQL node, the changes take effect immediately on all other MySQL servers attached to the cluster. Note that you may need to execute `FLUSH PRIVILEGES` on any SQL nodes with connected MySQL clients (or to disconnect and then reconnect the clients) in order for those clients to be able to see the changes in privileges.

Once `mysql_cluster_move_privileges` has been executed successfully, all MySQL user privileges are distributed across all connected MySQL Servers. MySQL Servers that join the cluster after this automatically participate in the privilege distribution.

`ndb_dist_priv.sql` also provides stored routines that can be used to verify that the privilege tables have been distributed successfully, and to perform other tasks.

For more information, see Distributed Privileges Using Shared Grant Tables.

- **Performance:** Added support for pushing down joins to the NDB kernel for parallel execution across the data nodes, which can speed up execution of joins across `NDB` tables by a factor of 20 or more in some cases. Some restrictions apply on the types of joins that can be optimized in this way; in particular, columns to be joined must use exactly the same data type, and cannot be any of the `BLOB` or `TEXT` types. In addition, columns to be joined must be part of a table index or primary key. Support for this feature can be enabled or disabled using the `ndb_join_pushdown` server system variable (enabled by default); see the description of this variable for more information and examples.

  As part of this improvement, the status variables `Ndb_pushed_queries_defined`, `Ndb_pushed_queries_dropped`, `Ndb_pushed_queries_executed`, and `Ndb_pushed_reads` also been introduced for monitoring purposes. You can also see whether a given join is pushed down using `EXPLAIN`. In addition, several new counters relating to push-down join performance have been added to the `counters` table in the `ndbinfo` database. For more information, see the descriptions of the status variables previously mentioned.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z