MySQL Cluster 7.3 Release Notes

Abstract

This document contains release notes for the changes in each release of MySQL Cluster that uses version 7.3 of the NDBCLUSTER storage engine.

Each MySQL Cluster NDB 7.3 release is based on a mainline MySQL Server 5.6 release and a particular version of the NDBCLUSTER 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 Cluster NDB 7.3.

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

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

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

The date mentioned with a release version is the date of the last revision control system changeset on which the release was based, not necessarily the date when the distribution packages were made available. The binaries are usually made available a few days after the date of the tagged changeset because building and testing all packages takes some time.

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.

Document generated on: 2014-08-20 (revision: 3962)

Table of Contents

Preface and Legal Notices

This document contains release notes for the changes in each release of MySQL Cluster that uses version 7.3 of the `NDBCLUSTER` storage engine.

Legal Notices

Changes in MySQL Cluster NDB 7.3.7 (5.6.20-ndb-7.3.7) (Not yet released)

MySQL Cluster NDB 7.3.7 is a new release of MySQL Cluster, based on MySQL Server 5.6 and including features from version 7.3 of the `NDB` storage engine, as well as fixing a number of recently discovered bugs in previous MySQL Cluster releases.

Obtaining MySQL Cluster NDB 7.3. MySQL Cluster NDB 7.3 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.3, see MySQL Cluster Development in MySQL Cluster NDB 7.3.

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

Functionality Added or Changed

- **Cluster API:** 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

- **Cluster API:** When two tables had different foreign keys with the same name, `ndb_restore` considered this a name conflict and failed to restore the schema. As a result of this fix, a slash character (/) is now expressly disallowed in foreign key names, and the naming format *parent_id*/*child_id*/*fk_name* is now enforced by the NDB API. (Bug #18824753)

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

  References: This bug 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)

- When the binlog injector thread commits an epoch to the binary log and this causes the log file to reach maximum size, it may need to rotate the binary log. The rotation is not performed until either all the committed transactions from all client threads are flushed to the binary log, or a maximum of 30 seconds has elapsed. In the case where all transactions were committed prior to the 30-second wait, it was possible for committed transactions from multiple client threads to belong to newer epochs than the latest epoch committed by the injector thread, causing the thread to deadlock with itself, and causing an unnecessary 30-second delay before breaking the deadlock. (Bug #18845822)

- Adding a foreign key failed with NDB Error 208 if the parent index was parent table's primary key, the primary key was not on the table's initial attributes, and the child table was not empty. (Bug #18825966)

- When an `NDB` table served as both the parent table and a child table for 2 different foreign keys having the same name, dropping the foreign key on the child table could cause the foreign key on the parent table to be dropped instead, leading to a situation in which it was impossible to drop the remaining foreign key. This situation can be modelled using the following `CREATE TABLE` statements:

```
CREATE TABLE parent (
    id INT NOT NULL,
    PRIMARY KEY (id)
) ENGINE=NDB;

CREATE TABLE child (
    id INT NOT NULL,
    parent_id INT,
    PRIMARY KEY (id),
    INDEX par_ind (parent_id),

    FOREIGN KEY (parent_id)
    REFERENCES parent(id)
) ENGINE=NDB;

CREATE TABLE grandchild (
    id INT,
    parent_id INT,
    INDEX par_ind (parent_id),

    FOREIGN KEY (parent_id)
    REFERENCES child(id)
) ENGINE=NDB;
```

  With the tables created as just shown, the issue occured when executing the statement `ALTER TABLE child DROP FOREIGN KEY parent_id`, because it was possible in some cases for `NDB` to drop the foreign key from the `grandchild` table instead. When this happened, any subsequent attempt to drop the foreign key from either the `child` or from the `grandchild` table failed. (Bug #18662582)

- `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 isuue 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)

- 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 bug is a regression of Bug #14220269.

- 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)

- 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)

- 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)

- The DEFINER column in the INFORMATION_SCHEMA.VIEWS table contained erroneous values for views contained in the ndbinfo information database. This could be seen in the result of a query such as SELECT TABLE_NAME, DEFINER FROM INFORMATION_SCHEMA.VIEWS WHERE TABLE_SCHEMA='ndbinfo'. (Bug #17018500)

- 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)

- **Disk Data:** Setting the undo buffer size used by `InitialLogFileGroup` to a value greater than that set by `SharedGlobalMemory` prevented data nodes from starting; the data nodes failed with Error 1504 `Out of logbuffer memory`. While the failure itself is expected behavior, the error message did not provide sufficient information to diagnose the actual source of the problem; now in such cases, a more specific error message `Out of logbuffer memory (specify smaller undo_buffer_size or increase SharedGlobalMemory)` is supplied. (Bug #11762867, Bug #55515)

- **Cluster 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)

- **Cluster API:** 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)

- **Cluster API:** 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)

- **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)

Changes in MySQL Cluster NDB 7.3.6 (5.6.19-ndb-7.3.6) (2014-07-11)

MySQL Cluster NDB 7.3.6 is a new release of MySQL Cluster, based on MySQL Server 5.6 and including features from version 7.3 of the `NDB` storage engine, as well as fixing a number of recently discovered bugs in previous MySQL Cluster releases.

Obtaining MySQL Cluster NDB 7.3. MySQL Cluster NDB 7.3 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.3, see MySQL Cluster Development in MySQL Cluster NDB 7.3.

This release also incorporates all bugfixes and changes made in previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.19 (see Changes in MySQL 5.6.19 (2014-05-30)).

Functionality Added or Changed

- **Cluster API:** 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

- **Cluster API:** When two tables had different foreign keys with the same name, `ndb_restore` considered this a name conflict and failed to restore the schema. As a result of this fix, a slash character (`/`) is now expressly disallowed in foreign key names, and the naming format *`parent_id`*/*`child_id`*/*`fk_name`* is now enforced by the NDB API. (Bug #18824753)

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

  References: This bug 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)

- When the binlog injector thread commits an epoch to the binary log and this causes the log file to reach maximum size, it may need to rotate the binary log. The rotation is not performed until either all the committed transactions from all client threads are flushed to the binary log, or a maximum of 30 seconds has elapsed. In the case where all transactions were committed prior to the 30-second wait, it was possible for committed transactions from multiple client threads to belong to newer epochs than the latest epoch committed by the injector thread, causing the thread to deadlock with itself, and causing an unnecessary 30-second delay before breaking the deadlock. (Bug #18845822)

- Adding a foreign key failed with NDB Error 208 if the parent index was parent table's primary key, the primary key was not on the table's initial attributes, and the child table was not empty. (Bug #18825966)

- When an `NDB` table served as both the parent table and a child table for 2 different foreign keys having the same name, dropping the foreign key on the child table could cause the foreign key on the parent table to be dropped instead, leading to a situation in which it was impossible to drop the remaining foreign key. This situation can be modelled using the following `CREATE TABLE` statements:

```
CREATE TABLE parent (
    id INT NOT NULL,
    PRIMARY KEY (id)
) ENGINE=NDB;

CREATE TABLE child (
    id INT NOT NULL,
    parent_id INT,
    PRIMARY KEY (id),
    INDEX par_ind (parent_id),

    FOREIGN KEY (parent_id)
    REFERENCES parent(id)
) ENGINE=NDB;

CREATE TABLE grandchild (
    id INT,
    parent_id INT,
    INDEX par_ind (parent_id),

    FOREIGN KEY (parent_id)
    REFERENCES child(id)
) ENGINE=NDB;
```

With the tables created as just shown, the issue occured when executing the statement `ALTER TABLE child DROP FOREIGN KEY parent_id`, because it was possible in some cases for `NDB` to drop the foreign key from the `grandchild` table instead. When this happened, any subsequent attempt to drop the foreign key from either the `child` or from the `grandchild` table failed. (Bug #18662582)

- `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 isuue 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)

- 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 bug is a regression of Bug #14220269.

- 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)

- 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)

- 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)

- The DEFINER column in the INFORMATION_SCHEMA.VIEWS table contained erroneous values for views contained in the ndbinfo information database. This could be seen in the result of a query such as SELECT TABLE_NAME, DEFINER FROM INFORMATION_SCHEMA.VIEWS WHERE TABLE_SCHEMA='ndbinfo'. (Bug #17018500)

- 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)

- **Disk Data:** Setting the undo buffer size used by InitialLogFileGroup to a value greater than that set by SharedGlobalMemory prevented data nodes from starting; the data nodes failed with Error 1504 Out of logbuffer memory. While the failure itself is expected behavior, the error message did not provide sufficient information to diagnose the actual source of the problem; now in such cases, a more specific error message Out of logbuffer memory (specify smaller undo_buffer_size or increase SharedGlobalMemory) is supplied. (Bug #11762867, Bug #55515)

- **Cluster 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)

- **Cluster API:** 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)

- **Cluster API:** 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)
- **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)

Changes in MySQL Cluster NDB 7.3.5 (5.6.17-ndb-7.3.5) (2014-04-07)

MySQL Cluster NDB 7.3.5 is a new release of MySQL Cluster, based on MySQL Server 5.6 and including features from version 7.3 of the `NDB` storage engine, as well as fixing a number of recently discovered bugs in previous MySQL Cluster releases.

Obtaining MySQL Cluster NDB 7.3. MySQL Cluster NDB 7.3 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.3, see MySQL Cluster Development in MySQL Cluster NDB 7.3.

This release also incorporates all bugfixes and changes made in previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.17 (see Changes in MySQL 5.6.17 (2014-03-27)).

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 Cluster release series, have now been removed. (Bug #11746486, Bug #26673)
- 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)

- 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)

- 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 Cluster NDB 7.3.2. (Bug #18296810)

- -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.

- 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)

- The block threads managed by the multi-threading 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 MySQL 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)

- An issue found when compiling the MySQL Cluster software for Solaris platforms could lead to problems when using `ThreadConfig` on such systems. (Bug #18181656)

- When creating a table with foreign key referencing an index in another table, it sometimes appeared possible to create the foreign key even if the order of the columns in the indexes did not match, due to the fact that an appropriate error was not always returned internally. This fix improves the error used internally to work in most cases; however, it is still possible for this situation to occur in the event that the parent index is a unique index. (Bug #18094360)

- Updating parent tables of foreign keys used excessive scan resources and so required unusually high settings for `MaxNoOfLocalScans` and `MaxNoOfConcurrentScans`. (Bug #18082045)

- Dropping a nonexistent foreign key on an `NDB` table (using, for example, `ALTER TABLE`) appeared to succeed. Now in such cases, the statement fails with a relevant error message, as expected. (Bug #17232212)

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

- **Replication:** Log rotation events could cause `group_relay_log_pos` to be moved forward incorrectly within a group. This meant that, when the transaction was retried, or if the SQL thread was stopped in the middle of a transaction following one or more log rotations (such that the transaction or group spanned multiple relay log files), part or all of the group was silently skipped.

  This issue has been addressed by correcting a problem in the logic used to avoid touching the coordinates of the SQL thread when updating the log position as part of a relay log rotation whereby it was possible to update the SQL thread's coordinates when not using a multi-threaded slave, even in the middle of a group. (Bug #18482854)

- **Cluster Replication:** A slave in MySQL 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.

- **Cluster API:** 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)

- **Cluster API:** Refactoring that was performed in MySQL Cluster NDB 7.3.4 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 bug was introduced by Bug #17647637.

- **Cluster API:** 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)

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

- **Cluster API:** When `Dictionary::dropTable()` attempted (as a normal part of its internal operations) to drop an index used by a foreign key constraint, the drop failed. Now in such cases, invoking `dropTable()` causes all foreign keys on the table to be dropped, whether this table acts as a parent table, child table, or both.

  This issue did not affect dropping of indexes using SQL statements. (Bug #18069680)

References: See also Bug #17591531.

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

Changes in MySQL Cluster NDB 7.3.4 (5.6.15-ndb-7.3.4) (2014-02-06)

MySQL Cluster NDB 7.3.4 is a new release of MySQL Cluster, based on MySQL Server 5.6 and including features from version 7.3 of the `NDB` storage engine, as well as fixing a number of recently discovered bugs in previous MySQL Cluster releases.

Obtaining MySQL Cluster NDB 7.3. MySQL Cluster NDB 7.3 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.3, see MySQL Cluster Development in MySQL Cluster NDB 7.3.

This release also incorporates all bugfixes and changes made in previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.15 (see Changes in MySQL 5.6.15 (2013-12-03)).

Bugs Fixed

- **Packaging:** 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)

- **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)

- **Cluster API:** `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.

- Interrupting a drop of a foreign key could cause the underlying table to become corrupt. (Bug #18041636)

- 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)

  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)

- 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)

- 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)

- 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.

- 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)

- 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)
- Trying to drop an index used by a foreign key constraint caused data node failure. Now in such cases, the statement used to perform the drop fails. (Bug #17591531)
- 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)
- 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)
- **Cluster API:** Compilation of example NDB API program files failed due to missing include directives. (Bug #17672846, Bug #70759)
- **Cluster API:** 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)
- **Cluster API:** An application, having opened two distinct instances of `Ndb_cluster_connection`, attempted to use the second connection object to send signals to itself, but these signals were blocked until the destructor was explicitly called for that connection object. (Bug #17626525)

  References: This bug is a regression of Bug #16595838.
- **ndbmemcache:** A memcached server running the NDB engine could crash after being disconnected from a cluster. (Bug #14055851)

Changes in MySQL Cluster NDB 7.3.3 (5.6.14-ndb-7.3.3) (2013-11-18)

MySQL Cluster NDB 7.3.3 is a new release of MySQL Cluster, based on MySQL Server 5.6 and including features from version 7.3 of the `NDB` storage engine, as well as fixing a number of recently discovered bugs in previous MySQL Cluster releases.

Obtaining MySQL Cluster NDB 7.3. MySQL Cluster NDB 7.3 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.3, see MySQL Cluster Development in MySQL Cluster NDB 7.3.

This release also incorporates all bugfixes and changes made in previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.14 (see Changes in MySQL 5.6.14 (2013-09-20)).

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)

- The MySQL Cluster Auto-Installer now generates a `my.cnf` file for each `mysqld` in the cluster before starting it. For more information, see Using the MySQL Cluster Auto-Installer. (Bug #16994782)

- `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 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:** Portions of the documentation specific to MySQL Cluster and the `NDB` storage engine were not included when installing from RPMs. (Bug #16303451)

- 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)

- It was not possible to start MySQL Cluster processes created by the Auto-Installer on a Windows host running freeSSHd. (Bug #17269626)

- `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)

- `ALTER ONLINE TABLE ... REORGANIZE PARTITION` failed when run against a table having or using a reference to a foreign key. (Bug #17036744, Bug #69619)

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

- `TUPKEYREQ` signals are used to read data from the tuple manager block (`DBTUP`), and are used for all types of data access, especially for scans which read many rows. A TUPKEYREQ specifies a series of 'columns' to be read, which can be either single columns in a specific table, or pseudocolumns, two of which—`READ_ALL` and `READ_PACKED`—are aliases to read all columns in a table, or some subset of these columns. Pseudocolumns are used by modern NDB API applications as they require less space in the `TUPKEYREQ` to specify columns to be read, and can return the data in a more compact (packed) format.

  This fix moves the creation and initialization of on-stack Signal objects to only those pseudocolumn reads which need to `EXECUTE_DIRECT` to other block instances, rather than for every read. In addition, the size of an on-stack signal is now varied to suit the requirements of each pseudocolumn, so that only reads of the `INDEX_STAT` pseudocolumn now require initialization (and 3KB memory each time this is performed). (Bug #17009502)

- A race condition could sometimes occur when trying to lock receive threads to cores. (Bug #17009393)
- `RealTimeScheduler` did not work correctly with data nodes running `ndbmtd`. (Bug #16961971)
- Results from joins using a `WHERE` with an `ORDER BY ... DESC` clause were not sorted properly; the `DESC` keyword in such cases was effectively ignored. (Bug #16999886, Bug #69528)
- 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)
- The Windows error *`ERROR_FILE_EXISTS`* was not recognized by `NDB`, which treated it as an unknown error. (Bug #16970960)
- 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)
- Dropping a column, which was not itself a foreign key, from an `NDB` table having foreign keys failed with **`ER_TABLE_DEF_CHANGED`**. (Bug #16912989)
- 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.
- `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)
- 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)
- 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.

- 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 multi-threading 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)

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

- 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)

- Setting `lower_case_table_names` to 1 or 2 on Windows systems caused `ALTER TABLE ... ADD FOREIGN KEY` statements against tables with names containing uppercase letters to fail with Error 155, `No such table: '(null)'`. (Bug #14826778, Bug #67354)

- 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)

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

- 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)

- 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)

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

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

- `ABORT BACKUP` in the `ndb_mgm` client (see Commands in the MySQL 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)

- 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)

- 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.

- **Cluster 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)

- **Cluster Replication:** Replaying a binary log that had been written by a `mysqld` from a MySQL Server distribution (and from not a MySQL Cluster distribution), and that contained DML statements, on a MySQL Cluster SQL node could lead to failure of the SQL node. (Bug #16742250)

- **Cluster API:** 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:** 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:** 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.

Changes in MySQL Cluster NDB 7.3.2 (5.6.11-ndb-7.3.2) (2013-06-18, General Availability)

MySQL Cluster NDB 7.3.2 is the first GA release of MySQL Cluster, based on MySQL Server 5.6 and including new features in version 7.3 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous MySQL Cluster releases.

Obtaining MySQL Cluster NDB 7.3. MySQL Cluster NDB 7.3 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL Cluster NDB 7.3, see MySQL Cluster Development in MySQL Cluster NDB 7.3.

This release also incorporates all bugfixes and changes made in previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.11 (see Changes in MySQL 5.6.11 (2013-04-18)).

Functionality Added or Changed

- **ndbmemcache:** ndbmemcache now supports values of types `DATE`, `TIME`, and `DATETIME` using microsecond precision. (Bug #16593493)

Bugs Fixed

- **Packaging:** The MySQL 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 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)

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

  References: This bug 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)

- 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)

- On Solaris SPARC platforms, batched key access execution of some joins could fail due to invalid memory access. (Bug #16818575)

- When 2 `NDB` tables had foreign key references to each other, it was necessary to drop the tables in the same order in which they were created. (Bug #16817928)

- The duplicate weedout algorithm introduced in MySQL 5.6 evaluates semi-joins such as subqueries using `IN`) by first performing a normal join between the outer and inner table which may create duplicates of rows form the outer (and inner) table and then removing any duplicate result rows from the outer table by comparing their primary key values. Problems could arise when `NDB` copied `VARCHAR` values using their maximum length, resulting in a binary key image which contained garbage past the actual lengths of the `VARCHAR` values, which meant that multiple instances of the same key were not binary-identical as assumed by the MySQL server.

  To fix this problem, `NDB` now zero-pads such values to the maximum length of the column so that copies of the same key are treated as identical by the weedout process. (Bug #16744050)

- `DROP DATABASE` failed to work correctly when executed against a database containing `NDB` tables joined by foreign key constraints (and all such tables being contained within this database), leaving these tables in place while dropping the remaining tables in the database and reporting failure. (Bug #16692652, Bug #69008)

- The `CLUSTERLOG` command (see Commands in the MySQL Cluster Management Client) caused `ndb_mgm` to crash on Solaris SPARC systems. (Bug #16723708)

- When using `firstmatch=on` with the `optimizer_switch` system variable, pushed joins could return too many rows. (Bug #16664035)

- A variable used by the batched key access implementation was not initialized by `NDB` as expected. This could cause a “batch full” condition to be reported after only a single row had been batched, effectively disabling batching altogether and leading to an excessive number of round trips between `mysqld` and `NDB`. (Bug #16485658)

- 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)

- 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)

- Setting `foreign_key_checks = 0` had no effect on the handling of `NDB` tables. Now, doing so causes such checks of foreign key constraints to be suspended—that is, has the same effect on `NDB` tables as it has on `InnoDB` tables. (Bug #14095855, Bug #16286309)

  References: See also Bug #16286164.

- **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)

- **Cluster Replication:** The implicit `FLUSH LOGS` performed as part of binary log rotation could deadlock and time out when using a sufficiently small value for `max_binlog_size`. (Bug #16884594)

- **Cluster API:** 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. (Bug #16834030)

- **Cluster API:** 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)

- **ClusterJ:** ClusterJ now provides partial support for MySQL date and time data types with fractional seconds (see Date and Time Type Overview), as shown in the following table:

ClusterJ Class	MySQL Data Types	Precision
`java.util.Date`	`DATETIME`	millisecond
	`TIME`	millisecond
	`TIMESTAMP`	millisecond
`java.sql.Date`	`DATE`	day
`java.sql.Timestamp`	`DATETIME`	millisecond
	`TIMESTAMP`	millisecond
`java.sql.Time`	`TIME`	second

  (Bug #16593510)

Changes in MySQL Cluster NDB 7.3.1 (5.6.10-ndb-7.3.1) (2013-04-17, Development Milestone)

MySQL Cluster NDB 7.3.1 is a new Developer Milestone release of MySQL Cluster, based on MySQL Server 5.6 and previewing new features under development for version 7.3 of the `NDB` storage engine, as well as fixing a number of recently discovered bugs in previous releases.

Obtaining MySQL Cluster NDB 7.3. MySQL Cluster NDB 7.3 source code and binaries can be obtained from http://dev.mysql.com/downloads/cluster/.

This release also incorporates all bugfixes and changes made in previous MySQL Cluster releases, as well as all bugfixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.10 (see Changes in MySQL 5.6.10 (2013-02-05, General Availability)).

Based on MySQL Server 5.6

- **Important Change:** MySQL Cluster SQL nodes are now based on MySQL Server 5.6. For information about feature additions and other changes made in MySQL 5.6, see What Is New in MySQL 5.6.

  The `mysqld` binary provided with MySQL Cluster NDB 7.3.1 is based on MySQL Server 5.6.10, and includes all MySQL Server 5.6 feature enhancements and bug fixes found in that release; see Changes in MySQL 5.6.10 (2013-02-05, General Availability), for information about these.

MySQL Cluster GUI Configuration Wizard

- **Important Change:** The MySQL Cluster distribution now includes a browser-based graphical configuration wizard that assists the user in configuring and deploying a MySQL Cluster. This deployment can consist of an arbitrary number of nodes (within certain limits) on the user machine only, or include nodes distributed on a local network. The wizard can be launched from the command line (using the `ndb_setup` utility now included in the binary distribution) or a desktop file browser.

  For more information about this tool, see The MySQL Cluster Auto-Installer.

Support for Foreign Key Constraints

- **Important Change:** MySQL Cluster now supports foreign key constraints between NDB tables, including support for CASCADE, SET NULL, and RESTRICT and NO ACTION reference options for DELETE and UPDATE actions. (MySQL currently does not support SET DEFAULT.)

  MySQL requires generally that all child and parent tables in foreign key relationships employ the same storage engine; thus, to use foreign keys with MySQL Cluster tables, the child and parent table must each use the NDB storage engine. (It is not possible, for example, for a foreign key on an NDB table to reference an index of an InnoDB table.)

  Note that MySQL Cluster tables that are explicitly partitioned by KEY or LINEAR KEY may contain foreign key references or be referenced by foreign keys (or both). This is unlike the case with InnoDB tables that are user partitioned, which may not have any foreign key relationships.

  You can create an NDB table having a foreign key reference on another NDB table using CREATE TABLE ... [CONSTRAINT] FOREIGN KEY ... REFERENCES. A child table's foreign key definitions can be seen in the output of SHOW CREATE TABLE; you can also obtain information about foreign keys by querying the INFORMATION_SCHEMA.KEY_COLUMN_USAGE table.

  FOREIGN KEY Constraints, provides general information about foreign key support in MySQL. For more information about the syntax supported by MySQL for foreign keys, see Using FOREIGN KEY Constraints.

NoSQL Connector for JavaScript (Node.js)

- **Cluster API:** MySQL Cluster NDB 7.3 includes support for JavaScript applications written against Node.js with MySQL Cluster and MySQL Server as a data store. The Connector for JavaScript provides a domain object model similar in many ways to that employed by ClusterJ (see The ClusterJ API and Data Object Model) and can be used with either of two backend adapters: the ndb adapter, which uses the NDB API to provide high-performance native access to MySQL Cluster; and the mysql-js adapter, which uses the MySQL Server and the node-mysql driver available from https://github.com/felixge/node-mysql/ .

  The Connector for JavaScript is included with the MySQL Cluster distribution, and contains setup programs which can assist you with installation of the connector. You must have Node.js and MySQL Cluster installed prior to running the setup scripts. The node-mysql driver is also required for the mysql-js Node.js adapter; you can install this using the package management tool included with Node.js. For more information, see MySQL NoSQL Connector for JavaScript.

Functionality Added or Changed

- **Important Change:** The behavior of and values used for the TCP_RCV_BUF_SIZE and TCP_SND_BUF_SIZE TCP configuration parameters have been improved. Formerly, the default values for these parameters were 70080 and 71540, respectively—which it was later found could lead to excessive timeouts in some circumstances—with the minimum for each of them being 1. Now, the default and recommended value is 0 for both TCP_RCV_BUF_SIZE and TCP_SND_BUF_SIZE, which allows the operating system or platform to choose the send or receive buffer size for TCP sockets. (Bug #14554519)

  References: See also Bug #14168828.

- **Cluster API:** 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 MySQL 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)

- 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

- **Important Note; Cluster Replication:** Setting `binlog_row_image=minimal` caused MySQL Cluster replication conflict resolution to fail.

  To fix this problem, support for this variable is disabled in the `NDB` storage engine such that setting it or the corresponding `--binlog-row-image` server option has no effect for `NDB` tables. (Bug #16316828)

- **API:** `mysqld` failed to respond when `mysql_shutdown()` was invoked from a C application, or `mysqladmin shutdown` was run from the command line. (Bug #14849574)

- When an update of an `NDB` table changes the primary key (or part of the primary key), the operation is executed as a delete plus an insert. In some cases, the initial read operation did not retrieve all column values required by the insert, so that another read was required. This fix ensures that all required column values are included in the first read in such cases, which saves the overhead of an additional read operation. (Bug #16614114)

- Pushed joins executed when `optimizer_switch='batched_key_access=on'` was also in use returned incorrect results. (Bug #16437431)

- Selecting from the `INFORMATION_SCHEMA.KEY_COLUMN_USAGE` table while using tables with foreign keys caused `mysqld` to crash. (Bug #16246874, Bug #68224)

- 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.

- 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)

- Purging the binary logs could sometimes cause `mysqld` to crash. (Bug #15854719)

- **ndbmemcache:** When using a large `values` table, `memcached` failed to store flags correctly or not at all in `NDB`, even when it had been configured to, as follows:

  - In MySQL Cluster NDB 7.2.4 and earlier, the flags value was never stored for long values, even if configured to do so.

  - In MySQL Cluster NDB 7.2.6 and later, non-zero flags values were correctly stored for long values, but a value of zero was not stored.

  In addition, because some clients (such as Spymemcached for Java) use the flags field, this fix also enables storage of flags for the values table by default. (Bug #14088078)

Release Series Changelogs: MySQL Cluster NDB 7.3

This section contains unified changelog information for the MySQL Cluster NDB 7.3 release series.

For changelogs covering individual MySQL Cluster NDB 7.3 releases, see MySQL Cluster Release Notes.

For general information about features added in MySQL Cluster NDB 7.3, see MySQL Cluster Development History.

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

Changes in the MySQL Cluster NDB 7.3 Series

This section contains unified change history highlights for all MySQL Cluster releases based on version 7.3 of the `NDBCLUSTER` storage engine through MySQL Cluster NDB 7.3.7. Included are all changelog entries in the categories *MySQL Cluster*, *Disk Data*, and *Cluster API*.

For an overview of features that were added in MySQL Cluster NDB 7.3, see MySQL Cluster Development History.

Changes in MySQL Cluster NDB 7.3.7 (5.6.20-ndb-7.3.7)

Version 5.6.20-ndb-7.3.7 has no changelog entries.

Changes in MySQL Cluster NDB 7.3.6 (5.6.19-ndb-7.3.6)

Functionality Added or Changed

- **Cluster API:** 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

- **Cluster API:** When two tables had different foreign keys with the same name, `ndb_restore` considered this a name conflict and failed to restore the schema. As a result of this fix, a slash character (`/`) is now expressly disallowed in foreign key names, and the naming format *`parent_id`*/*`child_id`*/*`fk_name`* is now enforced by the NDB API. (Bug #18824753)

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

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

- When the binlog injector thread commits an epoch to the binary log and this causes the log file to reach maximum size, it may need to rotate the binary log. The rotation is not performed until either all the committed transactions from all client threads are flushed to the binary log, or a maximum of 30 seconds has elapsed. In the case where all transactions were committed prior to the 30-second wait, it was possible for committed transactions from multiple client threads to belong to newer epochs than the latest epoch committed by the injector thread, causing the thread to deadlock with itself, and causing an unnecessary 30-second delay before breaking the deadlock. (Bug #18845822)

- 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)

- Adding a foreign key failed with NDB Error 208 if the parent index was parent table's primary key, the primary key was not on the table's initial attributes, and the child table was not empty. (Bug #18825966)

- When an `NDB` table served as both the parent table and a child table for 2 different foreign keys having the same name, dropping the foreign key on the child table could cause the foreign key on the parent table to be dropped instead, leading to a situation in which it was impossible to drop the remaining foreign key. This situation can be modelled using the following `CREATE TABLE` statements:

```
CREATE TABLE parent (
    id INT NOT NULL,
    PRIMARY KEY (id)
) ENGINE=NDB;

CREATE TABLE child (
    id INT NOT NULL,
    parent_id INT,
    PRIMARY KEY (id),
    INDEX par_ind (parent_id),

    FOREIGN KEY (parent_id)
    REFERENCES parent(id)
) ENGINE=NDB;

CREATE TABLE grandchild (
    id INT,
    parent_id INT,
    INDEX par_ind (parent_id),

    FOREIGN KEY (parent_id)
    REFERENCES child(id)
) ENGINE=NDB;
```

  With the tables created as just shown, the issue occured when executing the statement `ALTER TABLE child DROP FOREIGN KEY parent_id`, because it was possible in some cases for `NDB` to drop the foreign key from the `grandchild` table instead. When this happened, any subsequent attempt to drop the foreign key from either the `child` or from the `grandchild` table failed. (Bug #18662582)

- 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 bug is a regression of Bug #14220269.

- 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 isuue 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)

- 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)

- The `DEFINER` column in the `INFORMATION_SCHEMA.VIEWS` table contained erroneous values for views contained in the `ndbinfo` information database. This could be seen in the result of a query such as `SELECT TABLE_NAME, DEFINER FROM INFORMATION_SCHEMA.VIEWS WHERE TABLE_SCHEMA='ndbinfo'`. (Bug #17018500)

- 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)

- **Disk Data:** Setting the undo buffer size used by `InitialLogFileGroup` to a value greater than that set by `SharedGlobalMemory` prevented data nodes from starting; the data nodes failed with Error 1504 `Out of logbuffer memory`. While the failure itself is expected behavior, the error message did not provide sufficient information to diagnose the actual source of the problem; now in such cases, a more specific error message `Out of logbuffer memory (specify smaller undo_buffer_size or increase SharedGlobalMemory)` is supplied. (Bug #11762867, Bug #55515)

- **Cluster API:** 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)

- **Cluster API:** 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)

Changes in MySQL Cluster NDB 7.3.5 (5.6.17-ndb-7.3.5)

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 Cluster release series, have now been removed. (Bug #11746486, Bug #26673)

- 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)

- 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)

- 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 Cluster NDB 7.3.2. (Bug #18296810)

- -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.

- 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)

- The block threads managed by the multi-threading 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)

- An issue found when compiling the MySQL Cluster software for Solaris platforms could lead to problems when using ThreadConfig on such systems. (Bug #18181656)

- The ndb_mgm client START BACKUP command (see Commands in the MySQL 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)

- Updating parent tables of foreign keys used excessive scan resources and so required unusually high settings for MaxNoOfLocalScans and MaxNoOfConcurrentScans. (Bug #18082045)

- When creating a table with foreign key referencing an index in another table, it sometimes appeared possible to create the foreign key even if the order of the columns in the indexes did not match, due to the fact that an appropriate error was not always returned internally. This fix improves the error used internally to work in most cases; however, it is still possible for this situation to occur in the event that the parent index is a unique index. (Bug #18094360)

- Dropping a nonexistent foreign key on an `NDB` table (using, for example, `ALTER TABLE`) appeared to succeed. Now in such cases, the statement fails with a relevant error message, as expected. (Bug #17232212)
- Data nodes running `ndbmtd` could stall while performing an online upgrade of a MySQL Cluster containing a great many tables from a version prior to NDB 7.2.5 to version 7.2.5 or later. (Bug #16693068)
- **Cluster API:** 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)
- **Cluster API:** Refactoring that was performed in MySQL Cluster NDB 7.3.4 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 bug was introduced by Bug #17647637.
- **Cluster API:** 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)

- **Cluster API:** When Dictionary::dropTable() attempted (as a normal part of its internal operations) to drop an index used by a foreign key constraint, the drop failed. Now in such cases, invoking dropTable() causes all foreign keys on the table to be dropped, whether this table acts as a parent table, child table, or both.

  This issue did not affect dropping of indexes using SQL statements. (Bug #18069680)

  References: See also Bug #17591531.

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

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

Changes in MySQL Cluster NDB 7.3.4 (5.6.15-ndb-7.3.4)

Bugs Fixed

- **Packaging:** 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)

- **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)

- **Cluster API:** 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.

- Interrupting a drop of a foreign key could cause the underlying table to become corrupt. (Bug #18041636)

- 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)

  References: See also Bug #17934985.

- 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)

- 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.

- 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)

- 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)

- 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.

- 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)

- 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)

- Trying to drop an index used by a foreign key constraint caused data node failure. Now in such cases, the statement used to perform the drop fails. (Bug #17591531)

- 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)

- 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)

- **Cluster API:** 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)

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

- **Cluster API:** An application, having opened two distinct instances of `Ndb_cluster_connection`, attempted to use the second connection object to send signals to itself, but these signals were blocked until the destructor was explicitly called for that connection object. (Bug #17626525)

  References: This bug is a regression of Bug #16595838.

Changes in MySQL Cluster NDB 7.3.3 (5.6.14-ndb-7.3.3)

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)

- The MySQL Cluster Auto-Installer now generates a `my.cnf` file for each `mysqld` in the cluster before starting it. For more information, see Using the MySQL Cluster Auto-Installer. (Bug #16994782)

- `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 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:** Portions of the documentation specific to MySQL Cluster and the `NDB` storage engine were not included when installing from RPMs. (Bug #16303451)

- 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)

- It was not possible to start MySQL Cluster processes created by the Auto-Installer on a Windows host running freeSSHd. (Bug #17269626)

- `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)

- `TUPKEYREQ` signals are used to read data from the tuple manager block (`DBTUP`), and are used for all types of data access, especially for scans which read many rows. A TUPKEYREQ specifies a series of 'columns' to be read, which can be either single columns in a specific table, or pseudocolumns, two of which—`READ_ALL` and `READ_PACKED`—are aliases to read all columns in a table, or some subset of these columns. Pseudocolumns are used by modern NDB API applications as they require less space in the `TUPKEYREQ` to specify columns to be read, and can return the data in a more compact (packed) format.

  This fix moves the creation and initialization of on-stack Signal objects to only those pseudocolumn reads which need to `EXECUTE_DIRECT` to other block instances, rather than for every read. In addition, the size of an on-stack signal is now varied to suit the requirements of each pseudocolumn, so that only reads of the `INDEX_STAT` pseudocolumn now require initialization (and 3KB memory each time this is performed). (Bug #17009502)

- A race condition could sometimes occur when trying to lock receive threads to cores. (Bug #17009393)

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

- `ALTER ONLINE TABLE ... REORGANIZE PARTITION` failed when run against a table having or using a reference to a foreign key. (Bug #17036744, Bug #69619)

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

- Results from joins using a `WHERE` with an `ORDER BY ... DESC` clause were not sorted properly; the `DESC` keyword in such cases was effectively ignored. (Bug #16999886, Bug #69528)

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

- 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)

- Dropping a column, which was not itself a foreign key, from an `NDB` table having foreign keys failed with **`ER_TABLE_DEF_CHANGED`**. (Bug #16912989)

- 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)

- 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)

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

- 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.

- `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)

- 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)

- 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.

- 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)

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

- When the available job buffers for a given thread fell below the critical threshold, the internal multi-threading 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)

- Setting `lower_case_table_names` to 1 or 2 on Windows systems caused `ALTER TABLE ... ADD FOREIGN KEY` statements against tables with names containing uppercase letters to fail with Error 155, `No such table: '(null)'`. (Bug #14826778, Bug #67354)

- 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)

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

- 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)

- 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)

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

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

- 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 MySQL 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.

- **Cluster API:** 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)

Changes in MySQL Cluster NDB 7.3.2 (5.6.11-ndb-7.3.2)

Bugs Fixed

- **Packaging:** The MySQL 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 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)

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

- 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)

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

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

- 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)

- 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)

- When 2 `NDB` tables had foreign key references to each other, it was necessary to drop the tables in the same order in which they were created. (Bug #16817928)

- 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)

- On Solaris SPARC platforms, batched key access execution of some joins could fail due to invalid memory access. (Bug #16818575)

- The duplicate weedout algorithm introduced in MySQL 5.6 evaluates semi-joins such as subqueries using `IN`) by first performing a normal join between the outer and inner table which may create duplicates of rows form the outer (and inner) table and then removing any duplicate result rows from the outer table by comparing their primary key values. Problems could arise when `NDB` copied `VARCHAR` values using their maximum length, resulting in a binary key image which contained garbage past the actual lengths of the `VARCHAR` values, which meant that multiple instances of the same key were not binary-identical as assumed by the MySQL server.

  To fix this problem, `NDB` now zero-pads such values to the maximum length of the column so that copies of the same key are treated as identical by the weedout process. (Bug #16744050)

- The `CLUSTERLOG` command (see Commands in the MySQL Cluster Management Client) caused `ndb_mgm` to crash on Solaris SPARC systems. (Bug #16723708)

- `DROP DATABASE` failed to work correctly when executed against a database containing `NDB` tables joined by foreign key constraints (and all such tables being contained within this database), leaving these tables in place while dropping the remaining tables in the database and reporting failure. (Bug #16692652, Bug #69008)

- When using `firstmatch=on` with the `optimizer_switch` system variable, pushed joins could return too many rows. (Bug #16664035)

- A variable used by the batched key access implementation was not initialized by `NDB` as expected. This could cause a “batch full” condition to be reported after only a single row had been batched, effectively disabling batching altogether and leading to an excessive number of round trips between `mysqld` and `NDB`. (Bug #16485658)

- 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)

- 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)

- Setting `foreign_key_checks = 0` had no effect on the handling of `NDB` tables. Now, doing so causes such checks of foreign key constraints to be suspended—that is, has the same effect on `NDB` tables as it has on `InnoDB` tables. (Bug #14095855, Bug #16286309)

  References: See also Bug #16286164.

- **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)

- **Cluster API:** 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. (Bug #16834030)

- **Cluster API:** 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)

Changes in MySQL Cluster NDB 7.3.1 (5.6.10-ndb-7.3.1)

Based on MySQL Server 5.6

- **Important Change:** MySQL Cluster SQL nodes are now based on MySQL Server 5.6. For information about feature additions and other changes made in MySQL 5.6, see What Is New in MySQL 5.6.

  The `mysqld` binary provided with MySQL Cluster NDB 7.3.1 is based on MySQL Server 5.6.10, and includes all MySQL Server 5.6 feature enhancements and bug fixes found in that release; see Changes in MySQL 5.6.10 (2013-02-05, General Availability), for information about these.

MySQL Cluster GUI Configuration Wizard

- **Important Change:** The MySQL Cluster distribution now includes a browser-based graphical configuration wizard that assists the user in configuring and deploying a MySQL Cluster. This deployment can consist of an arbitrary number of nodes (within certain limits) on the user machine only, or include nodes distributed on a local network. The wizard can be launched from the command line (using the `ndb_setup` utility now included in the binary distribution) or a desktop file browser.

  For more information about this tool, see The MySQL Cluster Auto-Installer.

Support for Foreign Key Constraints

- **Important Change:** MySQL Cluster now supports foreign key constraints between `NDB` tables, including support for `CASCADE`, `SET NULL`, and `RESTRICT` and `NO ACTION` reference options for `DELETE` and `UPDATE` actions. (MySQL currently does not support `SET DEFAULT`.)

  MySQL requires generally that all child and parent tables in foreign key relationships employ the same storage engine; thus, to use foreign keys with MySQL Cluster tables, the child and parent table must each use the `NDB` storage engine. (It is not possible, for example, for a foreign key on an `NDB` table to reference an index of an `InnoDB` table.)

  Note that MySQL Cluster tables that are explicitly partitioned by `KEY` or `LINEAR KEY` may contain foreign key references or be referenced by foreign keys (or both). This is unlike the case with `InnoDB` tables that are user partitioned, which may not have any foreign key relationships.

  You can create an `NDB` table having a foreign key reference on another `NDB` table using `CREATE TABLE ... [CONSTRAINT] FOREIGN KEY ... REFERENCES`. A child table's foreign key definitions can be seen in the output of `SHOW CREATE TABLE`; you can also obtain information about foreign keys by querying the `INFORMATION_SCHEMA.KEY_COLUMN_USAGE` table.

  `FOREIGN KEY` Constraints, provides general information about foreign key support in MySQL. For more information about the syntax supported by MySQL for foreign keys, see Using `FOREIGN KEY` Constraints.

NoSQL Connector for JavaScript (Node.js)

- **Cluster API:** MySQL Cluster NDB 7.3 includes support for JavaScript applications written against Node.js with MySQL Cluster and MySQL Server as a data store. The Connector for JavaScript provides a domain object model similar in many ways to that employed by ClusterJ (see The ClusterJ API and Data Object Model) and can be used with either of two backend adapters: the `ndb` adapter, which uses the NDB API to provide high-performance native access to MySQL Cluster; and the `mysql-js` adapter, which uses the MySQL Server and the `node-mysql` driver available from https://github.com/felixge/node-mysql/ .

The Connector for JavaScript is included with the MySQL Cluster distribution, and contains setup programs which can assist you with installation of the connector. You must have Node.js and MySQL Cluster installed prior to running the setup scripts. The `node-mysql` driver is also required for the `mysql-js` Node.js adapter; you can install this using the package management tool included with Node.js. For more information, see MySQL NoSQL Connector for JavaScript.

Functionality Added or Changed

- **Important Change:** The behavior of and values used for the `TCP_RCV_BUF_SIZE` and `TCP_SND_BUF_SIZE` TCP configuration parameters have been improved. Formerly, the default values for these parameters were 70080 and 71540, respectively—which it was later found could lead to excessive timeouts in some circumstances—with the minimum for each of them being 1. Now, the default and recommended value is 0 for both `TCP_RCV_BUF_SIZE` and `TCP_SND_BUF_SIZE`, which allows the operating system or platform to choose the send or receive buffer size for TCP sockets. (Bug #14554519)

  References: See also Bug #14168828.

- **Cluster API:** 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 MySQL 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)

- 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

- **API:** `mysqld` failed to respond when `mysql_shutdown()` was invoked from a C application, or `mysqladmin shutdown` was run from the command line. (Bug #14849574)

- When an update of an `NDB` table changes the primary key (or part of the primary key), the operation is executed as a delete plus an insert. In some cases, the initial read operation did not retrieve all column values required by the insert, so that another read was required. This fix ensures that all required column values are included in the first read in such cases, which saves the overhead of an additional read operation. (Bug #16614114)

- Pushed joins executed when `optimizer_switch='batched_key_access=on'` was also in use returned incorrect results. (Bug #16437431)

- Selecting from the `INFORMATION_SCHEMA.KEY_COLUMN_USAGE` table while using tables with foreign keys caused `mysqld` to crash. (Bug #16246874, Bug #68224)

- 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.

- 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)
- Purging the binary logs could sometimes cause `mysqld` to crash. (Bug #15854719)