MySQL NDB Cluster 7.6 Release Notes

Abstract

This document contains release notes for the changes in each release of MySQL NDB Cluster that uses version 7.6 of the `NDB` (`NDBCLUSTER`) storage engine.

Each NDB Cluster 7.6 release is based on a mainline MySQL Server release and a particular version of the `NDB` storage engine, as shown in the version string returned by executing `SELECT VERSION()` in the `mysql` client, or by executing the `ndb_mgm` client `SHOW` or `STATUS` command; for more information, see MySQL NDB Cluster 7.5 and NDB Cluster 7.6.

For general information about features added in NDB Cluster 7.6, see What is New in NDB Cluster 7.6. For a complete list of all bug fixes and feature changes in MySQL NDB Cluster, please refer to the changelog section for each individual NDB Cluster release.

For additional MySQL 5.7 documentation, see the MySQL 5.7 Reference Manual, which includes an overview of features added in MySQL 5.7 that are not specific to NDB Cluster (What Is New in MySQL 5.7), and discussion of upgrade issues that you may encounter for upgrades from MySQL 5.6 to MySQL 5.7 (Changes in MySQL 5.7). For a complete list of all bug fixes and feature changes made in MySQL 5.7 that are not specific to `NDB`, see MySQL 5.7 Release Notes.

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

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

For legal information, see the Legal Notices.

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

Document generated on: 2018-11-21 (revision: 16512)

Table of Contents

Preface and Legal Notices

This document contains release notes for the changes in each release of MySQL NDB Cluster that uses version 7.6 of the NDB storage engine.

Legal Notices

Documentation Accessibility

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

Access to Oracle Support

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

Changes in MySQL NDB Cluster 7.6.9 (5.7.24-ndb-7.6.9) (Not yet released, General Availability)

MySQL NDB Cluster 7.6.9 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 7.6. NDB Cluster 7.6 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 7.6, see What is New in NDB Cluster 7.6.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.24 (see Changes in MySQL 5.7.24 (2018-10-22, General Availability)).

Version 5.7.24-ndb-7.6.9 has no changelog entries, or they have not been published because the product version has not been released.

Changes in MySQL NDB Cluster 7.6.8 (5.7.24-ndb-7.6.8) (2018-10-23, General Availability)

MySQL NDB Cluster 7.6.8 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 7.6. NDB Cluster 7.6 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 7.6, see What is New in NDB Cluster 7.6.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.24 (see Changes in MySQL 5.7.24 (2018-10-22, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Performance:** This release introduces a number of significant improvements in the performance of scans; these are listed here:
  - Row checksums help detect hardware issues, but do so at the expense of performance. `NDB` now offers the possibility of disabling these by setting the new `ndb_row_checksum` server system variable to 0; doing this means that row checksums are not used for new or altered tables. This can have a significant impact (5 to 10 percent, in some cases) on performance for all types of queries. This variable is set to 1 by default, to provide compatibility with the previous behavior.
  - A query consisting of a scan can execute for a longer time in the LDM threads when the queue is not busy.
  - Previously, columns were read before checking a pushed condition; now checking of a pushed condition is done before reading any columns.
  - Performance of pushed joins should see significant improvement when using range scans as part of join execution.

Bugs Fixed

- **Packaging:** Expected NDB header files were in the `devel` RPM package instead of `libndbclient-devel`. (Bug #84580, Bug #26448330)
- **NDB Disk Data:** While restoring a local checkpoint, it is possible to insert a row that already exists in the database; this is expected behavior which is handled by deleting the existing row first, then inserting the new copy of that row. In some cases involving data on disk, `NDB` failed to delete the existing row. (Bug #91627, Bug #28341843)
- **NDB Client Programs:** Removed a memory leak in `NdbImportUtil::RangeList` that was revealed in ASAN builds. (Bug #91479, Bug #28264144)
- **MySQL NDB ClusterJ:** When a table containing a `BLOB` or a `TEXT` field was being queried with ClusterJ for a record that did not exist, an exception ("`The method is not valid in current blob state`") was thrown. (Bug #28536926)

- **MySQL NDB ClusterJ:** A `NullPointerException` was thrown when a full table scan was performed with ClusterJ on tables containing either a BLOB or a TEXT field. It was because the proper object initializations were omitted, and they have now been added by this fix. (Bug #28199372, Bug #91242)

- When copying deleted rows from a live node to a node just starting, it is possible for one or more of these rows to have a global checkpoint index equal to zero. If this happened at the same time that a full local checkpoint was started due to the undo log getting full, the `LCP_SKIP` bit was set for a row having GCI = 0, leading to an unplanned shutdown of the data node. (Bug #28372628)

- `ndbmtd` sometimes experienced a hang when exiting due to log thread shutdown. (Bug #28027150)

- When the `SUMA` kernel block receives a `SUB_STOP_REQ` signal, it executes the signal then replies with `SUB_STOP_CONF`. (After this response is relayed back to the API, the API is open to send more `SUB_STOP_REQ` signals.) After sending the `SUB_STOP_CONF`, SUMA drops the subscription if no subscribers are present, which involves sending multiple `DROP_TRIG_IMPL_REQ` messages to `DBTUP`. LocalProxy can handle up to 21 of these requests in parallel; any more than this are queued in the Short Time Queue. When execution of a `DROP_TRIG_IMPL_REQ` was delayed, there was a chance for the queue to become overloaded, leading to a data node shutdown with `Error in short time queue`.

  This issue is fixed by delaying the execution of the `SUB_STOP_REQ` signal if `DBTUP` is already handling `DROP_TRIG_IMPL_REQ` signals at full capacity, rather than queueing up the `DROP_TRIG_IMPL_REQ` signals. (Bug #26574003)

- Having a large number of deferred triggers could sometimes lead to job buffer exhaustion. This could occur due to the fact that a single trigger can execute many operations—for example, a foreign key parent trigger may perform operations on multiple matching child table rows—and that a row operation on a base table can execute multiple triggers. In such cases, row operations are executed in batches. When execution of many triggers was deferred—meaning that all deferred triggers are executed at pre-commit—the resulting concurrent execution of a great many trigger operations could cause the data node job buffer or send buffer to be exhausted, leading to failure of the node.

  This issue is fixed by limiting the number of concurrent trigger operations as well as the number of trigger fire requests outstanding per transaction.

  For immediate triggers, limiting of concurrent trigger operations may increase the number of triggers waiting to be executed, exhausting the trigger record pool and resulting in the error `Too many concurrently fired triggers (increase MaxNoOfFiredTriggers`. This can be avoided by increasing `MaxNoOfFiredTriggers`, reducing the user transaction batch size, or both. (Bug #22529864)

  References: See also: Bug #18229003, Bug #27310330.

- `ndbout` and `ndberr` became invalid after exiting from `mgmd_run()`, and redirecting to them before the next call to `mgmd_run()` caused a segmentation fault, during an `ndb_mgmd` service restart. This fix ensures that `ndbout` and `ndberr` remain valid at all times. (Bug #17732772, Bug #28536919)

- Running out of undo log buffer memory was reported using error 921 `Out of transaction memory ... (increase SharedGlobalMemory)`.

  This problem is fixed by introducing a new error code 923 `Out of undo buffer memory (increase UNDO_BUFFER_SIZE)`. (Bug #92125, Bug #28537319)

- When moving an `OperationRec` from the serial to the parallel queue, `Dbacc::startNext()` failed to update the `Operationrec::OP_ACC_LOCK_MODE` flag which is required to reflect the accumulated `OP_LOCK_MODE` of all previous operations in the parallel queue. This inconsistency in the ACC lock queues caused the scan lock takeover mechanism to fail, as it incorrectly concluded that a lock to take

over was not held. The same failure caused an assert when aborting an operation that was a member of such an inconsistent parallel lock queue. (Bug #92100, Bug #28530928)

- A data node failed during startup due to the arrival of a `SCAN_FRAGREQ` signal during the restore phase. This signal originated from a scan begun before the node had previously failed and which should have been aborted due to the involvement of the failed node in it. (Bug #92059, Bug #28518448)

- `DBTUP` sent the error `Tuple corruption detected` when a read operation attempted to read the value of a tuple inserted within the same transaction. (Bug #92009, Bug #28500861)

- False constraint violation errors could occur when executing updates on self-referential foreign keys. (Bug #91965, Bug #28486390)

  References: See also: Bug #90644, Bug #27930382.

- An `NDB` internal trigger definition could be dropped while pending instances of the trigger remained to be executed, by attempting to look up the definition for a trigger which had already been released. This caused unpredictable and thus unsafe behavior possibly leading to data node failure. The root cause of the issue lay in an invalid assumption in the code relating to determining whether a given trigger had been released; the issue is fixed by ensuring that the behavior of `NDB`, when a trigger definition is determined to have been released, is consistent, and that it meets expectations. (Bug #91894, Bug #28451957)

- In some cases, a workload that included a high number of concurrent inserts caused data node failures when using debug builds. (Bug #91764, Bug #28384750)

- During an initial node restart with disk data tables present and `TwoPassInitialNodeRestartCopy` enabled, `DBTUP` used an unsafe scan in disk order. Such scans are no longer employed in this case. (Bug #91724, Bug #28378227)

- Checking for old LCP files tested the table version, but this was not always dependable. Now, instead of relying on the table version, the check reagrds as invalid any LCP file having a `maxGCI` smaller than its `createGci`. (Bug #91637, Bug #28346565)

- In certain cases, a cascade update trigger was fired repeatedly on the same record, which eventually consumed all available concurrent operations, leading to Error 233 `Out of operation records in transaction coordinator (increase MaxNoOfConcurrentOperations)`. If `MaxNoOfConcurrentOperations` was set to a value sufficiently high to avoid this, the issue manifested as data nodes consuming very large amounts of CPU, very likely eventually leading to a timeout. (Bug #91472, Bug #28262259)

- Inserting a row into an `NDB` table having a self-referencing foreign key that referenced a unique index on the table other than the primary key failed with `ER_NO_REFERENCED_ROW_2`. This was due to the fact that `NDB` checked foreign key constraints before the unique index was updated, so that the constraint check was unable to use the index for locating the row. Now, in such cases, `NDB` waits until all unique index values have been updated before checking foreign key constraints on the inserted row. (Bug #90644, Bug #27930382)

  References: See also: Bug #91965, Bug #28486390.

- A connection string beginning with a slash (`/`) character is now rejected by `ndb_mgmd`.

  Our thanks to Daniël van Eeden for contributing this fix. (Bug #90582, Bug #27912892)

Changes in MySQL NDB Cluster 7.6.7 (5.7.23-ndb-7.6.7) (2018-07-27, General Availability)

MySQL NDB Cluster 7.6.7 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 7.6. NDB Cluster 7.6 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 7.6, see What is New in NDB Cluster 7.6.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.23 (see Changes in MySQL 5.7.23 (2018-07-27, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- As part of ongoing work to improve handling of local checkpoints and minimize the occurrence of issues relating to Error 410 (`REDO log overloaded`) during LCPs, `NDB` now implements adaptive LCP control, which moderates LCP scan priority and LCP writes according to redo log usage.

  The following changes have been made with regard to `NDB` configuration parameters:

  - The default value of `RecoveryWork` is increased from 50 to 60 (60% of storage reserved for LCP files).
  - The new `InsertRecoveryWork` parameter controls the percentage of `RecoveryWork` that is reserved for insert operations. The default value is 40 (40% of `RecoveryWork`); the minimum and maximum are 0 and 70, respectively. Increasing this value allows for more writes during an LCP, while limiting the total size of the LCP. Decreasing `InsertRecoveryWork` limits the number of writes used during an LCP, but results in more space being used.

  Implementing LCP control provides several benefits to `NDB` deployments. Clusters should now survive heavy loads using default configurations much better than previously, and it should now be possible to run them reliably on systems where the available disk space is approximately 2.1 times the amount of memory allocated to the cluster (that is, the amount of `DataMemory`) or more. It is important to bear in mind that the figure just cited does not account for disk space used by tables on disk.

  During load testing into a single data node with decreasing redo log sizes, it was possible to successfully load a very large quantity of data into NDB with 16GB reserved for the redo log while using no more than 50% of the redo log at any point in time.

  See What is New in NDB Cluster 7.6, as well as the descriptions of the parameters mentioned previously, for more information. (Bug #90709, Bug #27942974, Bug #27942583)

  References: See also: Bug #27926532, Bug #27169282.

Bugs Fixed

- **ndbinfo Information Database:** It was possible following a restart for (sometimes incomplete) fallback data to be used in populating the `ndbinfo.processes` table, which could lead to rows in this table with empty `process_name` values. Such fallback data is no longer used for this purpose. (Bug #27985339)

- **NDB Client Programs:** The executable file `host_info` is no longer used by `ndb_setup.py`. This file, along with its parent directory `share/mcc/host_info`, has been removed from the NDB Cluster distribution.

  In addition, installer code relating to an unused `dojo.zip` file was removed. (Bug #90743, Bug #27966467, Bug #27967561)

  References: See also: Bug #27621546.

- **MySQL NDB ClusterJ:** ClusterJ could not be built from source using JDK 9. (Bug #27977985)

- An `NDB` restore operation failed under the following conditions:

  - A data node was restarted
  - The local checkpoint for the fragment being restored used two `.ctl` files
  - The first of these `.ctl` files was the file in use
  - The LCP in question consisted of more than 2002 parts

  This happened because an array used in decompression of the `.ctl` file contained only 2002 elements, which led to memory being overwritten, since this data can contain up to 2048 parts. This issue is fixed by increasing the size of the array to accommodate 2048 elements. (Bug #28303209)

- Local checkpoints did not always handle `DROP TABLE` operations correctly. (Bug #27926532)

  References: This issue is a regression of: Bug #26908347, Bug #26968613.

- During the execution of `CREATE TABLE ... IF NOT EXISTS`, the internal `open_table()` function calls `ha_ndbcluster::get_default_num_partitions()` implicitly whenever `open_table()` finds out that the requested table already exists. In certain cases, `get_default_num_partitions()` was called without the associated `thd_ndb` object being initialized, leading to failure of the statement with MySQL error 157 `Could not connect to storage engine`. Now `get_default_num_partitions()` always checks for the existence of this `thd_ndb` object, and initializes it if necessary.

Changes in MySQL NDB Cluster 7.6.6 (5.7.22-ndb-7.6.6) (2018-05-31, General Availability)

MySQL NDB Cluster 7.6.6 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 7.6. NDB Cluster 7.6 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 7.6, see What is New in NDB Cluster 7.6.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.22 (see Changes in MySQL 5.7.22 (2018-04-19, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- When performing an NDB backup, the `ndbinfo.logbuffers` table now displays information regarding buffer usage by the backup process on each data node. This is implemented as rows reflecting two new log types in addition to `REDO` and `DD-UNDO`. One of these rows has the log type `BACKUP-DATA`, which shows the amount of data buffer used during backup to copy fragments to backup files. The other row has the log type `BACKUP-LOG`, which displays the amount of log buffer used during the backup to record changes made after the backup has started. One each of these `log_type` rows is shown in the `logbuffers` table for each data node in the cluster. Rows having these two log types are present in the table only while an NDB backup is currently in progress. (Bug #25822988)

- Added the `--logbuffer-size` option for `ndbd` and `ndbmtd`, for use in debugging with a large number of log messages. This controls the size of the data node log buffer; the default (32K) is intended for normal operations. (Bug #89679, Bug #27550943)

- The previously experimental shared memory (SHM) transporter is now supported in production. SHM works by transporting signals through writing them into memory, rather than on a socket. NDB already attempts to use SHM automatically between data nodes and API nodes sharing the same host. To enable explicit shared memory connections, set the `UseShm` SHM configuration parameter to `true`. When explicitly defining shared memory as the connection method, it is also necessary to identify the nodes at either end of the connection (`NodeId1` and `NodeId2` parameters), and to provide a shared memory key (`ShmKey`). In addition, to improve performance, it is also possible to set a spin time (`ShmSpinTime`) for the SHM transporter.

  Configuration of SHM is otherwise similar to that of the TCP transporter. NDB Cluster Shared-Memory Connections, provides additional information.

- The `SPJ` kernel block now takes into account when it is evaluating a join request in which at least some of the tables are used in inner joins. This means that `SPJ` can eliminate requests for rows or ranges as soon as it becomes known that a preceding request did not return any results for a parent row. This saves both the data nodes and the `SPJ` block from having to handle requests and result rows which never take part in a result row from an inner join.

  

  Note

  When upgrading from NDB 7.6.5 or earlier, you should be aware that this optimization depends on both API client and data node functionality, and so is not available until all of these have been upgraded.

- The poll receiver which `NDB` uses to read from sockets, execute messages from the sockets, and wake up other threads now offloads wakeup of other threads to a new thread that wakes up the other threads on request, and otherwise simply sleeps. This improves the scalability of a single cluster connection by keeping the receive thread from becoming overburdened by tasks including wakeup of other threads.

Bugs Fixed

- **Important Change; NDB Client Programs:** `ndb_top` ignored short forms of command-line options, and did not in all cases handle misformed long options correctly. As part of the fix for these issues, the

following changes have been made to command-line options used with `ndb_top` to bring them more into line with those used with other NDB Cluster and MySQL programs:

- The `--passwd` option is removed, and replaced by `--password` (short form `-p`).
- The short form `-t` for the `--port` option has been replaced by `-P`.
- The short form `-x` for the `--text` option has been replaced by `-t`.

(Bug #26907833)

References: See also: Bug #88236, Bug #20733646.

- **NDB Cluster APIs:** A previous fix for an issue, in which the failure of multiple data nodes during a partial restart could cause API nodes to fail, did not properly check the validity of the associated `NdbReceiver` object before proceeding. Now in such cases an invalid object triggers handling for invalid signals, rather than a node failure. (Bug #25902137)

  References: This issue is a regression of: Bug #25092498.

- **NDB Cluster APIs:** Incorrect results, usually an empty result set, were returned when `setBound()` was used to specify a `NULL` bound. This issue appears to have been caused by a problem in gcc, limited to cases using the old version of this method (which does not employ `NdbRecord`), and is fixed by rewriting the problematic internal logic in the old implementation. (Bug #89468, Bug #27461752)

- **NDB Cluster APIs:** Released NDB API objects are kept in one or more `Ndb_free_list` structures for later reuse. Each list also keeps track of all objects seized from it, and makes sure that these are eventually released back to it. In the event that the internal function `NdbScanOperation::init()` failed, it was possible for an `NdbApiSignal` already allocated by the `NdbOperation` to be leaked. Now in such cases, `NdbScanOperation::release()` is called to release any objects allocated by the failed `NdbScanOperation` before it is returned to the free list.

  This fix also handles a similar issue with `NdbOperation::init()`, where a failed call could also leak a signal. (Bug #89249, Bug #27389894)

- **NDB Client Programs:** `ndb_top` did not support a number of options common to most `NDB` programs. The following options are now supported:

  - `--defaults-file`
  - `--defaults-extra-file`
  - `--print-defaults`
  - `--no-defaults`
  - `--defaults-group-suffix`

  In addition, `ndb_top` now supports a `--socket` option (short form `-S`) for specifying a socket file to use for the connection. (Bug #86614, Bug #26236298)

- **MySQL NDB ClusterJ:** ClusterJ quit unexpectedly as there was no error handling in the `scanIndex()` function of the `ClusterTransactionImpl` class for a null returned to it internally by the `scanIndex()` method of the `ndbTransaction` class. (Bug #27297681, Bug #88989)

- In some circumstances, when a transaction was aborted in the `DBTC` block, there remained links to trigger records from operation records which were not yet reference-counted, but when such an operation record was released the trigger reference count was still decremented. (Bug #27629680)

- An NDB online backup consists of data, which is fuzzy, and a redo and undo log. To restore to a consistent state it is necessary to ensure that the log contains all of the changes spanning the capture of the fuzzy data portion and beyond to a consistent snapshot point. This is achieved by waiting for a GCI boundary to be passed after the capture of data is complete, but before stopping change logging and recording the stop GCI in the backup's metadata.

  At restore time, the log is replayed up to the stop GCI, restoring the system to the state it had at the consistent stop GCI. A problem arose when, under load, it was possible to select a GCI boundary which occurred too early and did not span all the data captured. This could lead to inconsistencies when restoring the backup; these could be be noticed as broken constraints or corrupted BLOB entries.

  Now the stop GCI is chosen is so that it spans the entire duration of the fuzzy data capture process, so that the backup log always contains all data within a given stop GCI. (Bug #27497461)

  References: See also: Bug #27566346.

- For NDB tables, when a foreign key was added or dropped as a part of a DDL statement, the foreign key metatdata for all parent tables referenced should be reloaded in the handler on all SQL nodes connected to the cluster, but this was done only on the mysqld on which the statement was executed. Due to this, any subsequent queries relying on foreign key metadata from the corresponding parent tables could return inconsistent results. (Bug #27439587)

  References: See also: Bug #82989, Bug #24666177.

- ANALYZE TABLE used excessive amounts of CPU on large, low-cardinality tables. (Bug #27438963)

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

  References: See also: Bug #28728603.

- A data node overload could in some situations lead to an unplanned shutdown of the data node, which led to all data nodes disconnecting from the management and nodes.

  This was due to a situation in which API_FAILREQ was not the last received signal prior to the node failure.

  As part of this fix, the transaction coordinator's handling of SCAN_TABREQ signals for an ApiConnectRecord in an incorrect state was also improved. (Bug #27381901)

  References: See also: Bug #47039, Bug #11755287.

- In a two-node cluster, when the node having the lowest ID was started using --nostart, API clients could not connect, failing with Could not alloc node id at HOST port PORT_NO: No free node id found for mysqld(API). (Bug #27225212)

- Changing MaxNoOfExecutionThreads without an initial system restart led to an unplanned data node shutdown. (Bug #27169282)

  References: This issue is a regression of: Bug #26908347, Bug #26968613.

- Race conditions sometimes occurred during asynchronous disconnection and reconnection of the transporter while other threads concurrently inserted signal data into the send buffers, leading to an unplanned shutdown of the cluster.

  As part of the work fixing this issue, the internal templating function used by the Transporter Registry when it prepares a send is refactored to use likely-or-unlikely logic to speed up execution, and to remove a number of duplicate checks for NULL. (Bug #24444908, Bug #25128512)

References: See also: Bug #20112700.

- `ndb_restore` sometimes logged data file and log file progress values much greater than 100%. (Bug #20989106)
- The internal function `BitmaskImpl::setRange()` set one bit fewer than specified. (Bug #90648, Bug #27931995)
- It was not possible to create an `NDB` table using `PARTITION_BALANCE` set to `FOR_RA_BY_LDM_X_2`, `FOR_RA_BY_LDM_X_3`, or `FOR_RA_BY_LDM_X_4`. (Bug #89811, Bug #27602352)

  References: This issue is a regression of: Bug #81759, Bug #23544301.
- Adding a `[tcp]` or `[shm]` section to the global configuration file for a cluster with multiple data nodes caused default TCP connections to be lost to the node using the single section. (Bug #89627, Bug #27532407)
- As a result of the reuse of code intended for send threads when performing an assist send, all of the local release send buffers were released to the global pool, which caused the intended level of the local send buffer pool to be ignored. Now send threads and assisting worker threads follow their own policies for maintaining their local buffer pools. (Bug #89119, Bug #27349118)
- When sending priority A signals, we now ensure that the number of pending signals is explicitly initialized. (Bug #88986, Bug #27294856)
- In a MySQL Cluster with one MySQL Server configured to write a binary log failure occurred when creating and using an `NDB` table with non-stored generated columns. The problem arose only when the product was built with debugging support. (Bug #86084, Bug #25957586)
- `ndb_restore --print_data --hex` did not print trailing 0s of `LONGVARBINARY` values. (Bug #65560, Bug #14198580)
- When the internal function `ha_ndbcluster::copy_fk_for_offline_alter()` checked dependent objects on a table from which it was supposed to drop a foreign key, it did not perform any filtering for foreign keys, making it possible for it to attempt retrieval of an index or trigger instead, leading to a spurious Error 723 (`No such table`).

Changes in MySQL NDB Cluster 7.6.5 (5.7.20-ndb-7.6.5) (2018-04-20, Development)

MySQL NDB Cluster 7.6.5 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 7.6. NDB Cluster 7.6 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 7.6, see What is New in NDB Cluster 7.6.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.20 (see Changes in MySQL 5.7.20 (2017-10-16, General Availability)).

Bugs Fixed

- **NDB Client Programs:** On Unix platforms, the Auto-Installer failed to stop the cluster when `ndb_mgmd` was installed in a directory other than the default. (Bug #89624, Bug #27531186)

- **NDB Client Programs:** The Auto-Installer did not provide a mechanism for setting the `ServerPort` parameter. (Bug #89623, Bug #27539823)
- An internal buffer being reused immediately after it had been freed could lead to an unplanned data node shutdown. (Bug #27622643)

  References: See also: Bug #28698831.
- Writing of LCP control files was not always done correctly, which in some cases could lead to an unplanned shutdown of the cluster.

  This fix adds the requirement that upgrades from NDB 7.6.4 (or earlier) to this release (or a later one) include initial node restarts. (Bug #26640486)
- Under certain conditions, data nodes restarted unnecessarily during execution of `ALTER TABLE... REORGANIZE PARTITION`. (Bug #25679639)

Changes in MySQL NDB Cluster 7.6.4 (5.7.20-ndb-7.6.4) (2018-01-31, Development Milestone 4)

MySQL NDB Cluster 7.6.4 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 7.6. NDB Cluster 7.6 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 7.6, see What is New in NDB Cluster 7.6.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.20 (see Changes in MySQL 5.7.20 (2017-10-16, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Incompatible Change; NDB Disk Data:** Due to changes in disk file formats, it is necessary to perform an `--initial` restart of each data node when upgrading to or downgrading from this release.
- **Important Change; NDB Disk Data:** NDB Cluster has improved node restart times and overall performance with larger data sets by implementing partial local checkpoints (LCPs). Prior to this release, an LCP always made a copy of the entire database.

  `NDB` now supports LCPs that write individual records, so it is no longer strictly necessary for an LCP to write the entire database. Since, at recovery, it remains necessary to restore the database fully, the strategy is to save one fourth of all records at each LCP, as well as to write the records that have changed since the last LCP.

  Two data node configuration parameters relating to this change are introduced in this release: `EnablePartialLcp` (default `true`, or enabled) enables partial LCPs. When partial LCPs are enabled, `RecoveryWork` controls the percentage of space given over to LCPs; it increases with the amount of work which must be performed on LCPs during restarts as opposed to that performed during normal operations. Raising this value causes LCPs during normal operations to require writing fewer records and so decreases the usual workload. Raising this value also means that restarts can take longer.

Important

Upgrading to NDB 7.6.4 or downgrading from this release requires purging then re-creating the `NDB` data node file system, which means that an initial restart of each data node is needed. An initial node restart still requires a complete LCP; a partial LCP is not used for this purpose.

A rolling restart or system restart is a normal part of an `NDB` software upgrade. When such a restart is performed as part of an upgrade to NDB 7.6.4 or later, any existing LCP files are checked for the presence of the LCP `sysfile`, indicating that the existing data node file system was written using NDB 7.6.4 or later. If such a node file system exists, but does not contain the `sysfile`, and if any data nodes are restarted without the `--initial` option, `NDB` causes the restart to fail with an appropriate error message. This detection can be performed only as part of an upgrade; it is not possible to do so as part of a downgrade to NDB 7.6.3 or earlier from a later release.

Exception: If there are no data node files—that is, in the event of a “clean” start or restart—using `--initial` is not required for a software upgrade, since this is already equivalent to an initial restart. (This aspect of restarts is unchanged from previous releases of NDB Cluster.)

This release also deprecates the data node configuration parameters `BackupDataBufferSize`, `BackupWriteSize`, and `BackupMaxWriteSize`; these are now subject to removal in a future NDB Cluster release. (Bug #27308632)

- **Important Change:** Added the `ndb_perror` utility for obtaining information about NDB Cluster error codes. This tool replaces `perror --ndb`; the `--ndb` option for `perror` is now deprecated and raises a warning when used; the option is subject to removal in a future NDB release.

  See `ndb_perror` — Obtain NDB Error Message Information, for more information. (Bug #81703, Bug #81704, Bug #23523869, Bug #23523926)

  References: See also: Bug #26966826, Bug #88086.

- **NDB Client Programs:** NDB Cluster Auto-Installer node configuration parameters as supported in the UI and accompanying documentation were in some cases hard coded to an arbitrary value, or were missing altogether. Configuration parameters, their default values, and the documentation have been better aligned with those found in release versions of the NDB Cluster software.

  One necessary addition to this task was implementing the mechanism which the Auto-Installer now provides for setting parameters that take discrete values. For example, the value of the data node parameter `Arbitration` must now be one of `Default`, `Disabled`, or `WaitExternal`.

  The Auto-Installer also now gets and uses the amount of disk space available to `NDB` on each host for deriving reasonable default values for configuration parameters which depend on this value.

  See The NDB Cluster Auto-Installer (NDB 7.5), for more information.

- **NDB Client Programs:** Secure connection support in the MySQL NDB Cluster Auto-Installer has been updated or improved in this release as follows:

  - Added a mechanism for setting SSH membership on a per-host basis.
  - Updated the Paramiko Python module to the most recent available version (2.6.1).

- Provided a place in the GUI for encrypted private key passwords, and discontinued use of hardcoded passwords.

Related enhancements implemented in the current release include the following:

- Discontinued use of cookies as a persistent store for NDB Cluster configuration information; these were not secure and came with a hard upper limit on storage. Now the Auto-Installer uses an encrypted file for this purpose.
- In order to secure data transfer between the web browser front end and the back end web server, the default communications protocol has been switched from HTTP to HTTPS.

See The NDB Cluster Auto-Installer (NDB 7.5), for more information.

- **MySQL NDB ClusterJ:** ClusterJ now supports CPU binding for receive threads through the setRecvThreadCPUids() and getRecvThreadCPUids() methods. Also, the receive thread activation threshold can be set and get with the setRecvThreadActivationThreshold() and getRecvThreadActivationThreshold() methods.
- It is now possible to specify a set of cores to be used for I/O threads performing offline multithreaded builds of ordered indexes, as opposed to normal I/O duties such as file I/O , compression , or decompression. “Offline” in this context refers to building of ordered indexes performed when the parent table is not being written to; such building takes place when an `NDB` cluster performs a node or system restart, or as part of restoring a cluster from backup using `ndb_restore --rebuild-indexes`.

  In addition, the default behaviour for offline index build work is modified to use all cores available to `ndbmtd`, rather limiting itself to the core reserved for the I/O thread. Doing so can improve restart and restore times and performance, availability, and the user experience.

  This enhancement is implemented as follows:

  1. The default value for `BuildIndexThreads` is changed from 0 to 128. This means that offline ordered index builds are now multithreaded by default.
  2. The default value for `TwoPassInitialNodeRestartCopy` is changed from `false` to `true`. This means that an initial node restart first copies all data from a “live” node to one that is starting —without creating any indexes—builds ordered indexes offline, and then again synchronizes its data with the live node, that is, synchronizing twice and building indexes offline between the two synchonizations. This causes an initial node restart to behave more like the normal restart of a node, and reduces the time required for building indexes.
  3. A new thread type (`idxbld`) is defined for the `ThreadConfig` configuration parameter, to allow locking of offline index build threads to specific CPUs.

  In addition, `NDB` now distinguishes the thread types that are accessible to “ThreadConfig” by the following two criteria:

  1. Whether the thread is an execution thread. Threads of types `main`, `ldm`, `recv`, `rep`, `tc`, and `send` are execution threads; thread types `io`, `watchdog`, and `idxbld` are not.
  2. Whether the allocation of the thread to a given task is permanent or temporary. Currently all thread types except `idxbld` are permanent.

  For additonal information, see the descriptions of the parameters in the Manual. (Bug #25835748, Bug #26928111)

- Added the `ODirectSyncFlag` configuration parameter for data nodes. When enabled, the data node treats all completed filesystem writes to the redo log as though they had been performed using `fsync`.

  

  Note

  This parameter has no effect if at least one of the following conditions is true:

  - `ODirect` is not enabled.
  - `InitFragmentLogFiles` is set to `SPARSE`.

  (Bug #25428560)

- Added the `ndbinfo.error_messages` table, which provides information about NDB Cluster errors, including error codes, status types, brief descriptions, and classifications. This makes it possible to obtain error information using SQL in the `mysql` client (or other MySQL client program), like this:

```
mysql> SELECT * FROM ndbinfo.error_messages WHERE error_code='321';
+------------+----------------------+-----------------+----------------------+
| error_code | error_description    | error_status    | error_classification |
+------------+----------------------+-----------------+----------------------+
|        321 | Invalid nodegroup id | Permanent error | Application error    |
+------------+----------------------+-----------------+----------------------+
1 row in set (0.00 sec)
```

  The query just shown provides equivalent information to that obtained by issuing `ndb_perror 321` or (now deprecated) `perror --ndb 321` on the command line. (Bug #86295, Bug #26048272)

- `ThreadConfig` now has an additional `nosend` parameter that can be used to prevent a `main`, `ldm`, `rep`, or `tc` thread from assisting the send threads, by setting this parameter to 1 for the given thread. By default, `nosend` is 0. It cannot be used with threads other than those of the types just listed.

- When executing a scan as a pushed join, all instances of `DBSPJ` were involved in the execution of a single query; some of these received multiple requests from the same query. This situation is improved by enabling a single SPJ request to handle a set of root fragments to be scanned, such that only a single SPJ request is sent to each `DBSPJ` instance on each node and batch sizes are allocated per fragment, the multi-fragment scan can obtain a larger total batch size, allowing for some scheduling optimizations to be done within `DBSPJ`, which can scan a single fragment at a time (giving it the total batch size allocation), scan all fragments in parallel using smaller sub-batches, or some combination of the two.

  Since the effect of this change is generally to require fewer SPJ requests and instances, performance of pushed-down joins should be improved in many cases.

- As part of work ongoing to optimize bulk DDL performance by ndbmtd, it is now possible to obtain performance improvements by increasing the batch size for the bulk data parts of DDL operations which process all of the data in a fragment or set of fragments using a scan. Batch sizes are now made configurable for unique index builds, foreign key builds, and online reorganization, by setting the respective data node configuration parameters listed here:

  - `MaxFKBuildBatchSize`: Maximum scan batch size used for building foreign keys.
  - `MaxReorgBuildBatchSize`: Maximum scan batch size used for reorganization of table partitions.
  - `MaxUIBuildBatchSize`: Maximum scan batch size used for building unique keys.

For each of the parameters just listed, the default value is 64, the minimum is 16, and the maximum is 512.

Increasing the appropriate batch size or sizes can help amortize inter-thread and inter-node latencies and make use of more parallel resources (local and remote) to help scale DDL performance.

- Formerly, the data node `LGMAN` kernel block processed undo log records serially; now this is done in parallel. The `rep` thread, which hands off undo records to local data handler (LDM) threads, waited for an LDM to finish applying a record before fetching the next one; now the `rep` thread no longer waits, but proceeds immediately to the next record and LDM.

  There are no user-visible changes in functionality directly associated with this work; this performance enhancement is part of the work being done in NDB 7.6 to improve undo long handling for partial local checkpoints.

- When applying an undo log the table ID and fragment ID are obtained from the page ID. This was done by reading the page from `PGMAN` using an extra `PGMAN` worker thread, but when applying the undo log it was necessary to read the page again.

  This became very inefficient when using `O_DIRECT` (see `ODirect`) since the page was not cached in the OS kernel.

  Mapping from page ID to table ID and fragment ID is now done using information the extent header contains about the table IDs and fragment IDs of the pages used in a given extent. Since the extent pages are always present in the page cache, no extra disk reads are required to perform the mapping, and the information can be read using existing `TSMAN` data structures.

- Added the `NODELOG DEBUG` command in the `ndb_mgm` client to provide runtime control over data node debug logging. `NODE DEBUG ON` causes a data node to write extra debugging information to its node log, the same as if the node had been started with `--verbose`. `NODELOG DEBUG OFF` disables the extra logging.

- Added the `LocationDomainId` configuration parameter for management, data, and API nodes. When using NDB Cluster in a cloud environment, you can set this parameter to assign a node to a given availability domain or availability zone. This can improve performance in the following ways:

  - If requested data is not found on the same node, reads can be directed to another node in the same availability domain.
  - Communication between nodes in different availability domains are guaranteed to use `NDB` transporters' WAN support without any further manual intervention.
  - The transporter's group number can be based on which availability domain is used, such that also SQL and other API nodes communicate with local data nodes in the same availability domain whenever possible.
  - The arbitrator can be selected from an availability domain in which no data nodes are present, or, if no such availability domain can be found, from a third availability domain.

  This parameter takes an integer value between 0 and 16, with 0 being the default; using 0 is the same as leaving `LocationDomainId` unset.

Bugs Fixed

- **Important Change:** The `--passwd` option for `ndb_top` is now deprecated. It is removed (and replaced with `--password`) in NDB 7.6.5. (Bug #88236, Bug #20733646)

References: See also: Bug #86615, Bug #26236320, Bug #26907833.

- **Replication:** With GTIDs generated for incident log events, MySQL error code 1590 (ER_SLAVE_INCIDENT) could not be skipped using the `--slave-skip-errors=1590` startup option on a replication slave. (Bug #26266758)

- **NDB Disk Data:** An `ALTER TABLE` that switched the table storage format between `MEMORY` and `DISK` was always performed in place for all columns. This is not correct in the case of a column whose storage format is inherited from the table; the column's storage type is not changed.

  For example, this statement creates a table `t1` whose column `c2` uses in-memory storage since the table does so implicitly:

  ```
  CREATE TABLE t1 (c1 INT PRIMARY KEY, c2 INT) ENGINE NDB;
  ```

  The `ALTER TABLE` statement shown here is expected to cause `c2` to be stored on disk, but failed to do so:

  ```
  ALTER TABLE t1 STORAGE DISK TABLESPACE ts1;
  ```

  Similarly, an on-disk column that inherited its storage format from the table to which it belonged did not have the format changed by `ALTER TABLE ... STORAGE MEMORY`.

  These two cases are now performed as a copying alter, and the storage format of the affected column is now changed. (Bug #26764270)

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

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

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

- **ndbinfo Information Database:** Counts of committed rows and committed operations per fragment used by some tables in `ndbinfo` were taken from the `DBACC` block, but due to the fact that commit signals can arrive out of order, transient counter values could be negative. This could happen if, for example, a transaction contained several interleaved insert and delete operations on the same row; in such cases, commit signals for delete operations could arrive before those for the corresponding insert operations, leading to a failure in `DBACC`.

  This issue is fixed by using the counts of committed rows which are kept in `DBTUP`, which do not have this problem. (Bug #88087, Bug #26968613)

- Errors in parsing `NDB_TABLE` modifiers could cause memory leaks. (Bug #26724559)

- Added `DUMP` code 7027 to facilitate testing of issues relating to local checkpoints. For more information, see DUMP 7027. (Bug #26661468)

- A previous fix intended to improve logging of node failure handling in the transaction coordinator included logging of transactions that could occur in normal operation, which made the resulting logs needlessly verbose. Such normal transactions are no longer written to the log in such cases. (Bug #26568782)

  References: This issue is a regression of: Bug #26364729.

- Due to a configuration file error, CPU locking capability was not available on builds for Linux platforms. (Bug #26378589)

- Some `DUMP` codes used for the `LGMAN` kernel block were incorrectly assigned numbers in the range used for codes belonging to `DBTUX`. These have now been assigned symbolic constants and numbers in the proper range (10001, 10002, and 10003). (Bug #26365433)

- Node failure handling in the `DBTC` kernel block consists of a number of tasks which execute concurrently, and all of which must complete before TC node failure handling is complete. This fix extends logging coverage to record when each task completes, and which tasks remain, includes the following improvements:

  - Handling interactions between GCP and node failure handling interactions, in which TC takeover causes GCP participant stall at the master TC to allow it to extend the current GCI with any transactions that were taken over; the stall can begin and end in different GCP protocol states. Logging coverage is extended to cover all scenarios. Debug logging is now more consistent and understandable to users.

  - Logging done by the `QMGR` block as it monitors duration of node failure handling duration is done more frequently. A warning log is now generated every 30 seconds (instead of 1 minute), and this now includes `DBDIH` block debug information (formerly this was written separately, and less often).

  - To reduce space used, `DBTC instance number:` is shortened to `DBTC number:`.

  - A new error code is added to assist testing.

  (Bug #26364729)

- During a restart, `DBLQH` loads redo log part metadata for each redo log part it manages, from one or more redo log files. Since each file has a limited capacity for metadata, the number of files which must be consulted depends on the size of the redo log part. These files are opened, read, and closed sequentially, but the closing of one file occurs concurrently with the opening of the next.

  In cases where closing of the file was slow, it was possible for more than 4 files per redo log part to be open concurrently; since these files were opened using the `OM_WRITE_BUFFER` option, more than 4 chunks of write buffer were allocated per part in such cases. The write buffer pool is not unlimited; if all redo log parts were in a similar state, the pool was exhausted, causing the data node to shut down.

  This issue is resolved by avoiding the use of `OM_WRITE_BUFFER` during metadata reload, so that any transient opening of more than 4 redo log files per log file part no longer leads to failure of the data node. (Bug #25965370)

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

- A join entirely within the materialized part of a semi-join was not pushed even if it could have been. In addition, `EXPLAIN` provided no information about why the join was not pushed. (Bug #88224, Bug #27022925)

  References: See also: Bug #27067538.

- When the duplicate weedout algorithm was used for evaluating a semi-join, the result had missing rows. (Bug #88117, Bug #26984919)

  References: See also: Bug #87992, Bug #26926666.

- A table used in a loose scan could be used as a child in a pushed join query, leading to possibly incorrect results. (Bug #87992, Bug #26926666)

- When representing a materialized semi-join in the query plan, the MySQL Optimizer inserted extra `QEP_TAB` and `JOIN_TAB` objects to represent access to the materialized subquery result. The join pushdown analyzer did not properly set up its internal data structures for these, leaving them uninitialized instead. This meant that later usage of any item objects referencing the materialized semi-join accessed an initialized `tableno` column when accessing a 64-bit `tableno` bitmask, possibly referring to a point beyond its end, leading to an unplanned shutdown of the SQL node. (Bug #87971, Bug #26919289)

- In some cases, a `SCAN_FRAGCONF` signal was received after a `SCAN_FRAGREQ` with a close flag had already been sent, clearing the timer. When this occurred, the next `SCAN_FRAGREF` to arrive caused time tracking to fail. Now in such cases, a check for a cleared timer is performed prior to processing the `SCAN_FRAGREF` message. (Bug #87942, Bug #26908347)

- While deleting an element in `Dbacc`, or moving it during hash table expansion or reduction, the method used (`getLastAndRemove()`) could return a reference to a removed element on a released page, which could later be referenced from the functions calling it. This was due to a change brought about by the implementation of dynamic index memory in NDB 7.6.2; previously, the page had always belonged to a single `Dbacc` instance, so accessing it was safe. This was no longer the case following the change; a page released in `Dbacc` could be placed directly into the global page pool where any other thread could then allocate it.

  Now we make sure that newly released pages in `Dbacc` are kept within the current `Dbacc` instance and not given over directly to the global page pool. In addition, the reference to a released page has been removed; the affected internal method now returns the last element by value, rather than by reference. (Bug #87932, Bug #26906640)

  References: See also: Bug #87987, Bug #26925595.

- The `DBTC` kernel block could receive a `TCRELEASEREQ` signal in a state for which it was unprepared. Now it such cases it responds with a `TCRELEASECONF` message, and subsequently behaves just as if the API connection had failed. (Bug #87838, Bug #26847666)

  References: See also: Bug #20981491.

- When a data node was configured for locking threads to CPUs, it failed during startup with `Failed to lock tid`.

  This was is a side effect of a fix for a previous issue, which disabled CPU locking based on the version of the available `glibc`. The specific `glibc` issue being guarded against is encountered only in response to an internal NDB API call (`Ndb_UnlockCPU()`) not used by data nodes (and which can be accessed only through internal API calls). The current fix enables CPU locking for data nodes and disables it only for the relevant API calls when an affected `glibc` version is used. (Bug #87683, Bug #26758939)

  References: This issue is a regression of: Bug #86892, Bug #26378589.

- `ndb_top` failed to build on platforms where the `ncurses` library did not define `stdscr`. Now these platforms require the `tinfo` library to be included. (Bug #87185, Bug #26524441)

- On completion of a local checkpoint, every node sends a `LCP_COMPLETE_REP` signal to every other node in the cluster; a node does not consider the LCP complete until it has been notified that all other nodes have sent this signal. Due to a minor flaw in the LCP protocol, if this message was delayed from another node other than the master, it was possible to start the next LCP before one or more nodes had completed the one ongoing; this caused problems with `LCP_COMPLETE_REP` signals from previous LCPs becoming mixed up with such signals from the current LCP, which in turn led to node failures.

  To fix this problem, we now ensure that the previous LCP is complete before responding to any `TCGETOPSIZEREQ` signal initiating a new LCP. (Bug #87184, Bug #26524096)

- NDB Cluster did not compile successfully when the build used `WITH_UNIT_TESTS=OFF`. (Bug #86881, Bug #26375985)

- Recent improvements in local checkpoint handling that use `OM_CREATE` to open files did not work correctly on Windows platforms, where the system tried to create a new file and failed if it already existed. (Bug #86776, Bug #26321303)

- A potential hundredfold signal fan-out when sending a `START_FRAG_REQ` signal could lead to a node failure due to a `job buffer full` error in start phase 5 while trying to perform a local checkpoint during a restart. (Bug #86675, Bug #26263397)

  References: See also: Bug #26288247, Bug #26279522.

- Compilation of NDB Cluster failed when using `-DWITHOUT_SERVER=1` to build only the client libraries. (Bug #85524, Bug #25741111)

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

Changes in MySQL NDB Cluster 7.6.3 (5.7.18-ndb-7.6.3) (2017-07-03, Development Milestone 3)

MySQL NDB Cluster 7.6.3 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 7.6. NDB Cluster 7.6 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 7.6, see What is New in NDB Cluster 7.6.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.18 (see Changes in MySQL 5.7.18 (2017-04-10, General Availability)).

- Packaging Notes
- Functionality Added or Changed
- Bugs Fixed

Packaging Notes

- `mysqladmin` was added to Docker/Minimal packages because it is needed by InnoDB Cluster. (Bug #25998285)

Functionality Added or Changed

- **Important Change; MySQL NDB ClusterJ:** The ClusterJPA plugin for OpenJPA is no longer supported by NDB Cluster, and has been removed from the distribution. (Bug #23563810)
- **NDB Replication:** Added the `--ndb-log-update-minimal` option for logging by `mysqld`. This option causes only primary key values to be written in the before image, and only changed columns in the after image. (Bug #24438868)
- **MySQL NDB ClusterJ:** A new automatic reconnection feature has been implemented to facilitate the handling of connectivity issues. The feature is enabled by setting a positive number for a new connection property, `com.mysql.clusterj.connection.autoreconnect.timeout`, which specifies the length of the timeout period in seconds. If a connectivity error occurs, ClusterJ attempts to reconnect the application to the NDB Cluster after the application closes all the sessions; if the application does not close all sessions within the timeout period, ClusterJ closes any open sections forcibly, and then attempts reconnection. See Error Handling and Reconnection for details.
- In some critical situations such as data node failure, it was possible for the volume of log messages produced to cause file system and other issues, which compounded the problem, due to the fact that these messages were logged synchronously using `stdout`. To keep this from happening, log messages from worker threads now use a log buffer instead, which is nonblocking, and thus much less likely to cause interference with other processes under such conditions. (Bug #24748843)
- Added the `--diff-default` option for `ndb_config`. This option causes the program to print only those parameters having values that differ from their defaults. (Bug #85831, Bug #25844166)
- Added the `ndb_top` program on unix-based platforms. This utility shows CPU load and usage information for an `NDB` data node, with periodic updates (each second, by default). The display is in text or color ASCII graph format; both formats can be displayed at the same time. It is also possible to disable color output for the graph.

  `ndb_top` connects to an NDB Cluster SQL node—that is, a MySQL Server—and for this reason must be able to connect as a MySQL user having the `SELECT` privilege on tables in the `ndbinfo` database.

  `ndb_top` is not currently available for Windows platforms.

  For more information, see `ndb_top` — View CPU usage information for NDB threads.

Bugs Fixed

- **Packaging:** Two missing dependencies were added to the `apt` packages:
  - The data node package requires `libclass-methodmaker-perl`
  - The auto-installer requires `python-paramiko`

  (Bug #85679, Bug #25799465)
- **NDB Disk Data:** If the tablespace for a disk table had been fully consumed when a node failed, and table rows were deleted and inserted—or updated with shrinking or expanding disk column values—while the node was unavailable, a subsequent restart could fail with error 1601 `Out of extents, tablespace full`. We prevent this from happening by reserving 4 percent of the tablespace for use during node starts. (Bug #25923125)
- **NDB Replication:** Added a check to stop an `NDB` replication slave when configuration as a multithreaded slave is detected (for example, if `slave_parallel_workers` is set to a nonzero value). (Bug #21074209)

- **NDB Cluster APIs:** The implementation method `NdbDictionary::NdbTableImpl::getColumn()`, used from many places in the NDB API where a column is referenced by name, has been made more efficient. This method used a linear search of an array of columns to find the correct column object, which could be inefficient for tables with many columns, and was detected as a significant use of CPU in customer applications. (Ideally, users should perform name-to-column object mapping, and then use column IDs or objects in method calls, but in practice this is not always done.) A less costly hash index implementation, used previously for the name lookup, is reinstated for tables having relatively many columns. (A linear search continues to be used for tables having fewer columns, where the difference in performance is neglible.) (Bug #24829435)

- **NDB Cluster APIs:** NDB error 631 is reclassified as the (temporary) node recovery error `Scan take over error, restart scan transaction`. This was previously exposed to applications as an internal (and permanent) error which provided no description. (Bug #86401, Bug #26116231)

- **MySQL NDB ClusterJ:** The JTie and NDB JTie tests were skipped when the unit tests for ClusterJ were being run. (Bug #26088583)

- **MySQL NDB ClusterJ:** Compilation for the tests for NDB JTie failed. It was due to how null references were handled, which has been corrected by this fix. (Bug #26080804)

- Backup `.log` files contained log entries for one or more extra fragments, due to an issue with filtering out changes logged by other nodes in the same node group. This resulted in a larger `.log` file and thus use of more resources than necessary; it could also cause problems when restoring, since backups from different nodes could interfere with one another while the log was being applied. (Bug #25891014)

- Memory exhaustion during fragment creation led to an unplanned shutdown of the cluster. This issue could be triggered by the addition of unique keys to a large number of columns at the same time. (Bug #25851801)

- When making the final write to a redo log file, it is expected that the next log file is already opened for writes, but this was not always the case with a slow disk, leading to node failure. Now in such cases `NDB` waits for the next file to be opened properly before attempting to write to it. (Bug #25806659)

- Data node threads can be bound to a single CPU or a set of CPUs, a set of CPUs being represented internally by `NDB` as a `SparseBitmask`. When attempting to lock to a set of CPUs, CPU usage was excessive due to the fact that the routine performing the locks used the `mt_thr_config.cpp::do_bind()` method, which looks for bits that are set over the entire theoretical range of the `SparseBitmask` (2^{32}-2, or 4294967294). This is fixed by using `SparseBitmask::getBitNo()`, which can be used to iterate over only those bits that are actually set, instead. (Bug #25799506)

- Setting `NoOfFragmentLogParts` such that there were more than 4 redo log parts per local data manager led to resource exhaustion and subsequent multiple data node failures. Since this is an invalid configuration, a check has been added to detect a configuration with more than 4 redo log parts per LDM, and reject it as invalid. (Bug #25333414)

- In certain cases, a failed `ALTER TABLE ... ADD UNIQUE KEY` statement could lead to SQL node failure. (Bug #24444878)

  References: This issue is a regression of: Bug #23089566.

- Error 240 is raised when there is a mismatch between foreign key trigger columns and the values supplied to them during trigger execution, but had no error message indicating the source of the problem. (Bug #23141739)

  References: See also: Bug #23068914, Bug #85857.

- If the number of LDM blocks was not evenly divisible by the number of TC/SPJ blocks, SPJ requests were not equally distributed over the available SPJ instances. Now a round-robin distribution is used to distribute SPJ requests across all available SPJ instances more effectively.

  As part of this work, a number of unused member variables have been removed from the class `Dbtc`. (Bug #22627519)

- `ALTER TABLE .. MAX_ROWS=0` can now be performed only by using a copying `ALTER TABLE` statement. Resetting `MAX_ROWS` to 0 can no longer be performed using `ALGORITHM=INPLACE`. (Bug #21960004)

- During a system restart, when a node failed due to having missed sending heartbeats, all other nodes reported only that another node had failed without any additional information. Now in such cases, the fact that heartbeats were missed and the ID of the node that failed to send heartbeats is reported in both the error log and the data node log. (Bug #21576576)

- Due to a previous issue with unclear separation between the optimize and execute phases when a query involved a `GROUP BY`, the join-pushable evaluator was not sure whether its optimized query execution plan was in fact pushable. For this reason, such grouped joins were always considered not pushable. It has been determined that the separation issue has been resolved by work already done in MySQL 5.6, and so we now remove this limitation. (Bug #86623, Bug #26239591)

- When deleting all rows from a table immediately followed by `DROP TABLE`, it was possible that the shrinking of the `DBACC` hash index was not ready prior to the drop. This shrinking is a per-fragment operation that does not check the state of the table. When a table is dropped, `DBACC` releases resources, during which the description of the fragment size and page directory is not consistent; this could lead to reads of stale pages, and undefined behavior.

  Inserting a great many rows followed by dropping the table should also have had such effects due to expansion of the hash index.

  To fix this problem we make sure, when a fragment is about to be released, that there are no pending expansion or shrinkage operations on this fragment. (Bug #86449, Bug #26138592)

- Some error messages still referred to `IndexMemory`, although that parameter has been deprecated. (Bug #86385, Bug #26107514)

- The internal function `execute_signals()` in `mt.cpp` read three section pointers from the signal even when none was passed to it. This was mostly harmless, although unneeded. When the signal read was the last one on the last page in the job buffer, and the next page in memory was not mapped or otherwise accessible, `ndbmtd` failed with an error. To keep this from occurring, this function now only reads section pointers that are actually passed to it. (Bug #86354, Bug #26092639)

- There was at most one attempt in Dbacc to remove hash index pages freed when a table was dropped. This meant that, for large partitions (32 pages or more) there were always some pages lost. Now all hash index pages are freed when table using them is dropped. (Bug #86247, Bug #26030894)

- When a query on an `NDB` table failed due to a foreign key constraint violation, no useful information about the foreign key was shown in the error message, which contained only the text `Unknown error code`. (Bug #86241, Bug #26029485, Bug #16371292)

  References: See also: Bug #16275684.

- The `ndb_show_tables` program `--unqualified` option did not work correctly when set to 0 (false); this should disable the option and so cause fully qualified table and index names to be printed in the output. (Bug #86017, Bug #25923164)

- When an NDB table with foreign key constraints is created, its indexes are created first, and then, during foreign key creation, these indexes are loaded into the NDB dictionary cache. When a CREATE TABLE statement failed due to an issue relating to foreign keys, the indexes already in the cache were not invalidated. This meant that any subsequent CREATE TABLE with any indexes having the same names as those in the failed statement produced inconsistent results. Now, in such cases, any indexes named in the failed CREATE TABLE are immediately invalidated from the cache. (Bug #85917, Bug #25882950)

- During a local checkpoint, the record size is obtained from the DBTUP kernel block. This record size remained in use until the LCP scan was completed, which made it possible for DBTUP to update the maximum record size on commit of an ALTER TABLE that added a column to the table, and which could lead to node failure during the LCP. Now the record size is fetched at a point where updating it does not lead to this condition. (Bug #85858, Bug #25860002)

- Attempting to execute ALTER TABLE ... ADD FOREIGN KEY when the key to be added had the name of an existing foreign key on the same table failed with the wrong error message. (Bug #85857, Bug #23068914)

- The node internal scheduler (in mt.cpp) collects statistics about its own progress and any outstanding work it is performing. One such statistic is the number of outstanding send bytes, collected in send_buffer::m_node_total_send_buffer_size. This information may later be used by the send thread scheduler, which uses it as a metric to tune its own send performance versus latency.

  In order to reduce lock contention on the internal send buffers, they are split into two thr_send_buffer parts, m_buffer and m_sending, each protected by its own mutex, and their combined size repesented by m_node_total_send_buffer_size.

  Investigation of the code revealed that there was no consistency as to which mutex was used to update m_node_total_send_buffer_size, with the result that there was no consurrency protection for this value. To avoid this, m_node_total_send_buffer_size is replaced with two values, m_buffered_size and m_sending_size, which keep separate track of the sizes of the two buffers. These counters are updated under the protection of two different mutexes protecting each buffer individually, and are now added together to obtain the total size.

  With concurrency control established, updates of the partial counts should now be correct, so that their combined value no longer accumulates errors over time. (Bug #85687, Bug #25800933)

- Enabled the use of short or packed short TRANSID_AI signals for sending results from DBSPJ back to the client API. (Bug #85545, Bug #25750355)

  References: See also: Bug #85525, Bug #25741170.

- The maximum BatchByteSize as sent in SCANREQ signals was not always set correctly to reflect a limited byte size available in the client result buffers. The result buffer size calculation has been modified such that the effective batch byte size accurately reflects the maximum that may be returned by data nodes to prevent a possible overflow of the result buffers. (Bug #85411, Bug #25703113)

- When compiling the NDB kernel with gcc version 6.0.0 or later, it is now built using -flifetime-dse=1. (Bug #85381, Bug #25690926)

Changes in MySQL NDB Cluster 7.6.2 (5.7.18-ndb-7.6.2) (2017-04-26, Development Milestone 2)

MySQL NDB Cluster 7.6.2 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 7.6. NDB Cluster 7.6 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 7.6, see What is New in NDB Cluster 7.6.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.18 (see Changes in MySQL 5.7.18 (2017-04-10, General Availability)).

- Platform-Specific Notes
- Functionality Added or Changed
- Bugs Fixed

Platform-Specific Notes

- **Solaris:** The minimum required version of Solaris is now Solaris 11 update 3, due to a dependency on system runtime libraries.
- **Solaris:** On Solaris, MySQL is now built with Developer Studio 12.5 instead of `gcc`. The binaries require the Developer Studio C/C++ runtime libraries to be installed. See here for how to install only the libraries:

  ```
  https://docs.oracle.com/cd/E60778_01/html/E60743/gozsu.html
  ```

Functionality Added or Changed

- **Incompatible Change; NDB Disk Data:** Due to changes in disk file formats, it is necessary to perform an `--initial` restart of each data node when upgrading to or downgrading from this release.
- **Important Change:** As part of an ongoing effort to simplify NDB Cluster configuration, memory for indexes is now allocated dynamically from `DataMemory`; the `IndexMemory` configuration parameter is now deprecated, and is subject to removal in a future release. Any memory which has been set for `IndexMemory` in the `config.ini` file is now automatically added to `DataMemory`. In addition, the default value for `DataMemory` has been increased to 98M, and the default for `IndexMemory` has been decreased to 0.

  In addition to simplifying configuration of `NDB`, a further benefit of these changes is that scaling up by increasing the number of LDM threads is no longer limited by having set an insufficiently large value for `IndexMemory`. Previously, it was sometimes the case that increasing the number of LDM threads could lead to index memory exhaustion while large amounts of `DataMemory` remained available.

  Because instances of the `DBACC` kernel block (responsible for hash index storage) now share memory with each one another as well as with `DBLQH` (the kernel block that acts as the local data manager), they can take advantage of the fact that scaling up does not increase `DataMemory` usage greatly, and make use of spare memory for indexes freely. (For more information about these kernel blocks, see The DBACC Block, and The DBLQH Block.) In other words, index memory is no longer a static quantity allocated to each DBACC instance only once, on startup of the cluster, but rather this resource can now be allocated and deallocated whenever conditions require it.

  Related changes which have been made as part of this work are listed here:

  - Several instances of `DataMemory` usage not related to storage of table data now use transaction memory instead.

For this reason, it may be necessary on some systems to increase `SharedGlobalMemory`. In addition, systems performing initial bulk loads of data using large transactions may need to break up large transactions into smaller ones.

- Data nodes now generate `MemoryUsage` events (see NDB Cluster Log Events) and write appropriate messages in the cluster log when resource usage reaches 99%, in addition to when it reaches 80%, 90%, or 100% as they did previously.
- `REPORT MEMORYUSAGE` and other commands which expose memory consumption now shows index memory consumption using a page size of 32K rather than 8K.
- `IndexMemory` is no longer one of the values displayed in the `ndbinfo.memoryusage` table's `memory_type` column.
- The `ndbinfo.resources` table now shows the `DISK_OPERATIONS` resource as `TRANSACTION_MEMORY`.

  The `RESERVED` resource has been removed.
- `IndexMemory` is no longer displayed in `ndb_config` output.

- **Performance:** A number of debugging statements and printouts in the sources for the `DBTC` and `DBLQH` kernel blocks, as well as in related code, were moved into debugging code or removed altogether. This is expected to result in an improvement of up to 10% in the performance of local data management and transaction coordinator threads in many common use cases.
- **NDB Cluster APIs; ndbinfo Information Database:** Added two tables to the `ndbinfo` information database. The `config_nodes` table provides information about nodes that are configured as part of a given NDB Cluster, such as node ID and process type. The `processes` table shows information about nodes currently connected to the cluster; this information includes the process name and system process ID, and service address. For each data node and SQL node, it also shows the process ID of the node's angel process.

  As part of the work done to implement the `processes` table, a new `set_service_uri()` method has been added to the NDB API.

  For more information, see The ndbinfo config_nodes Table, and The ndbinfo processes Table, as well as Ndb_cluster_connection::set_service_uri().
- **NDB Cluster APIs:** The system name of an NDB cluster is now visible in the `mysql` client as the value of the `Ndb_system_name` status variable, and can also be obtained by NDB API application using the `Ndb_cluster_connection::get_system_name()` method. The system name can be set using the `Name` parameter in the `[system]` section of the cluster configuration file.
- Added the `--query-all` option to `ndb_config`. This option acts much like the `--query` option except that `--query-all` (short form: `-a`) dumps configuration information for all attributes at one time. (Bug #60095, Bug #11766869)
- Previously, when one LDM thread experienced I/O lag, such as during a disk overload condition, it wrote to a local checkpoint more slowly—that is, it wrote in I/O lag mode. However, other LDM threads did not necessarily observe or conform to this state. To ensure that write speed for the LCP is reduced by all LDM threads when such a slowdown is encountered, `NDB` now tracks I/O lag mode globally, so that I/O lag state is reported as soon as at least one thread is writing in I/O lag mode, and thus all LDM threads are forced to write in lag mode while the lag condition persists. This reduction in write speed by other LDM instances should increase overall capacity, enabling the disk overload condition to be overcome more quickly in such cases than before.

- Added the `ndb_import` tool to facilitate the loading of CSV-formatted data, such as that produced by `SELECT INTO OUTFILE`, into an `NDB` table. `ndb_import` is intended to function much like `mysqlimport` or the `LOAD DATA INFILE` SQL statement, and supports many similar options for formatting of the data file. A connection to an `NDB` management server (`ndb_mgmd`) is required; there must be an otherwise unused `[api]` slot in the cluster's `config.ini` file for this purpose. In addition, the target database and table (created using the `NDB` storage engine) must already exist, and the name of the CSV file (less any file extension) must be the same as that of the target table. A running SQL node is needed for creating the target database and table, but is not required for `ndb_import` to function.

  For more information, see `ndb_import` — Import CSV Data Into NDB.

Bugs Fixed

- **Partitioning:** The output of `EXPLAIN PARTITIONS` displayed incorrect values in the `partitions` column when run on an explicitly partitioned `NDB` table having a large number of partitions.

  This was due to the fact that, when processing an `EXPLAIN` statement, `mysqld` calculates the partition ID for a hash value as (*`hash_value % number_of_partitions`*), which is correct only when the table is partitioned by `HASH`, since other partitioning types use different methods of mapping hash values to partition IDs. This fix replaces the partition ID calculation performed by `mysqld` with an internal `NDB` function which calculates the partition ID correctly, based on the table's partitioning type. (Bug #21068548)

  References: See also: Bug #25501895, Bug #14672885.

- **Microsoft Windows:** When collecting information about CPUs on Windows, the Auto-Installer counted only physical cores, unlike on other platforms, where it collects information about both physical and virtual cores. Now the CPU information obtained on Windows is the same as that provided on other platforms. (Bug #85209, Bug #25636423)

- **Solaris; ndbmemcache:** `ndbmemcache` was not built correctly on Solaris platforms when compiling NDB Cluster using Developer Studio. (Bug #85477, Bug #25730703)

- **Solaris; MySQL NDB ClusterJ:** ClusterJ was not built correctly on Solaris platforms when compiling NDB Cluster using Oracle Developer Studio. (Bug #25738510)

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

- **NDB Replication:** Execution of `CREATE TABLE` could in some cases cause the replication slave SQL thread to hang. (Bug #85015, Bug #25654833)

  References: This issue is a regression of: Bug #83676, Bug #25042101.

- When `ndb_report_thresh_binlog_epoch_slip` was enabled, an event buffer status message with `report_reason=LOW/ENOUGH_FREE_EVENTBUFFER` was printed in the logs when event buffer usage was high and then decreased to a lower level. This calculation was based on total allocated event buffer memory rather than the limit set by `ndb_eventbuffer_max_alloc`; it was also printed even when the event buffer had unlimited memory (`ndb_eventbuffer_max_alloc` = 0, the default), which could confuse users.

  This is fixed as follows:

  - The calculation of `ndb_eventbuffer_free_percent` is now based on `ndb_eventbuffer_max_alloc`, rather than the amount actually allocated.

  - When `ndb_eventbuffer_free_percent` is set and `ndb_eventbuffer_max_alloc` is equal to 0, event buffer status messages using `report_reason=LOW/ENOUGH_FREE_EVENTBUFFER` are no longer printed.

  - When `ndb_report_thresh_binlog_epoch_slip` is set, an event buffer status message showing `report_reason=BUFFERED_EPOCHS_OVER_THRESHOLD` is written each 10 seconds (rather than every second) whenever this is greater than the threshold.

  (Bug #25726723)

- A bulk update is executed by reading records and executing a transaction on the set of records, which is started while reading them. When transaction initialization failed, the transaction executor function was subsequently unaware that this had occurred, leading to SQL node failures. This issue is fixed by providing appropriate error handling when attempting to initialize the transaction. (Bug #25476474)

  References: See also: Bug #20092754.

- CPU usage of the data node's main thread by the `DBDIH` master block as the end of a local checkpoint could approach 100% in certain cases where the database had a very large number of fragment replicas. This is fixed by reducing the frequency and range of fragment queue checking during an LCP. (Bug #25443080)

- Execution of an online `ALTER TABLE ... REORGANIZE PARTITION` statement on an `NDB` table having a primary key whose length was greater than 80 bytes led to restarting of data nodes, causing the reorganization to fail. (Bug #25152165)

- Multiple data node failures during a partial restart of the cluster could cause API nodes to fail. This was due to expansion of an internal object ID map by one thread, thus changing its location in memory, while another thread was still accessing the old location, leading to a segmentation fault in the latter thread.

  The internal `map()` and `unmap()` functions in which this issue arose have now been made thread-safe. (Bug #25092498)

  References: See also: Bug #25306089.

- The planned shutdown of an NDB Cluster having more than 10 data nodes was not always performed gracefully. (Bug #20607730)

- Dropped `TRANS_AI` signals that used the long signal format were not handled by the `DBTC` kernel block. (Bug #85606, Bug #25777337)

  References: See also: Bug #85519, Bug #27540805.

- Iproved pushed join handling by eliminating unneeded `FLUSH_AI` attributes that passed an empty row to the `DBSPJ` kernel block, when a row should be passed to the SPJ API only; this reduces the set of `AttrInfo` projections that must be executed in order to produce the result. This also makes it possible to employ packed `TRANSID_AI` signals when delivering SPJ API results, which is more efficient. (Bug #85525, Bug #25741170)

  References: See also: Bug #85545, Bug #25750355.

- Use of the long signal format (introduced in NDB 6.4) for an incoming `TRANSID_AI` message is supported by the `BACKUP`, `DBTC`, `DBLQH`, `SUMA`, `DBSPJ`, and `DBUTIL` NDB kernel blocks, but the `DBTUP` block produced long signals only when sending to `DPSPJ` or `DBUTIL`, and otherwise sent a series of short signals instead. Now `DBTUP` uses long signals for such messages whenever the receiving block supports this optimization. (Bug #85519, Bug #25740805)

- To prevent a scan from returning more rows, bytes, or both than the client has reserved buffers for, the DBTUP kernel block reports the size of the TRANSID_AI it has sent to the client in the TUPKEYCONF signal it sends to the requesting DBLQH block. DBLQH is aware of the maximum batch size available for the result set, and terminates the scan batch if this has been exceeded.

  The DBSPJ block's FLUSH_AI attribute allows DBTUP to produce two TRANSID_AI results from the same row, one for the client, and one for DBSPJ, which is needed for key lookups on the joined tables. The size of both of these were added to the read length reported by the DBTUP block, which caused the controlling DBLQH block to believe that it had consumed more of the available maximum batch size than was actually the case, leading to premature termination of the scan batch which could have a negative impact on performance of SPJ scans. To correct this, only the actual read length part of an API request is now reported in such cases. (Bug #85408, Bug #25702850)

- Data node binaries for Solaris 11 built using Oracle Developer Studio 12.5 on SPARC platforms failed with bus errors. (Bug #85390, Bug #25695818)

- During the initial phase of a scan request, the DBTC kernel block sends a series of DIGETNODESREQ signals to the DBDIH block in order to obtain dictionary information for each fragment to be scanned. If DBDIH returned DIGETNODESREF, the error code from that signal was not read, and Error 218 Out of LongMessageBuffer was always returned instead. Now in such cases, the error code from the DIGETNODESREF signal is actually used. (Bug #85225, Bug #25642405)

- If the user attempts to invoke ndb_setup.py while the Auto-Installer is still running—for example, after closing the terminal in which it was started and later opening a new terminal and invoking it in the new one—the program fails with the error Web server already running, which is expected behavior. In such cases, the mcc.pid file must first be removed prior to restarting the Auto-Installer (also expected behavior). Now when the program fails for this reason, the location of mcc.pid is included in the error message to simplify this task. (Bug #85169, Bug #25611093)

- The planned shutdown of a data node after one or more data nodes in the same node group had failed was not always performed correctly. (Bug #85168, Bug #25610703)

- There existed the possibility of a race condition between schema operations on the same database object originating from different SQL nodes; this could occur when one of the SQL nodes was late in releasing its metadata lock on the affected schema object or objects in such a fashion as to appear to the schema distribution coordinator that the lock release was acknowledged for the wrong schema change. This could result in incorrect application of the schema changes on some or all of the SQL nodes or a timeout with repeated waiting max ### sec for distributing... messages in the node logs due to failure of the distribution protocol. (Bug #85010, Bug #25557263)

  References: See also: Bug #24926009.

- When a foreign key was added to or dropped from an NDB table using an ALTER TABLE statement, the parent table's metadata was not updated, which made it possible to execute invalid alter operations on the parent afterwards.

  Until you can upgrade to this release, you can work around this problem by running SHOW CREATE TABLE on the parent immediately after adding or dropping the foreign key; this statement causes the table's metadata to be reloaded. (Bug #82989, Bug #24666177)

- Transactions on NDB tables with cascading foreign keys returned inconsistent results when the query cache was also enabled, due to the fact that mysqld was not aware of child table updates. This meant that results for a later SELECT from the child table were fetched from the query cache, which at that point contained stale data.

This is fixed in such cases by adding all children of the parent table to an internal list to be checked by `NDB` for updates whenever the parent is updated, so that `mysqld` is now properly informed of any updated child tables that should be invalidated from the query cache. (Bug #81776, Bug #23553507)

Changes in MySQL NDB Cluster 7.6.1 (5.7.17-ndb-7.6.1) (Not released, Development Milestone 1)

MySQL NDB Cluster 7.6.1 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 7.6. NDB Cluster 7.6.1 was an internal testing release only, and was not released to the public.

For an overview of changes made in NDB Cluster 7.6, see What is New in NDB Cluster.

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.17 (see Changes in MySQL 5.7.17 (2016-12-12, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **NDB Disk Data:** A new file format is introduced in this release for NDB Disk Data tables. The new format provides a mechanism whereby each Disk Data table can be uniquely identified without reusing table IDs. This is intended to help resolve issues with page and extent handling that were visible to the user as problems with rapid creating and dropping of Disk Data tables, and for which the old format did not provide a ready means to fix.

  The new format is now used whenever new undo log file groups and tablespace data files are created. Files relating to existing Disk Data tables continue to use the old format until their tablespaces and undo log file groups are re-created. *Important*: The old and new formats are not compatible and so cannot be employed for different data files or undo log files that are used by the same Disk Data table or tablespace.

  To avoid problems relating to the old format, you should re-create any existing tablespaces and undo log file groups when upgrading. You can do this by performing an initial restart of each data node (that is, using the `--initial` option) as part of the upgrade process. Since the current release is a pre-GA Developer release, this initial node restart is optional for now, but *you should expect it—and prepare for it now—to be mandatory in GA versions of NDB 7.6*.

  If you are using Disk Data tables, a downgrade from *any* NDB 7.6 release to any NDB 7.5 or earlier release requires restarting data nodes with `--initial` as part of the downgrade process, due to the fact that NDB 7.5 and earlier releases cannot read the new Disk Data file format.

  For more information, see Upgrading and Downgrading NDB Cluster.

Bugs Fixed

- **Packaging:** NDB Cluster Auto-Installer RPM packages for SLES 12 failed due to a dependency on `python2-crypto` instead of `python-pycrypto`. (Bug #25399608)

- **NDB Disk Data:** Stale data from NDB Disk Data tables that had been dropped could potentially be included in backups due to the fact that disk scans were enabled for these. To prevent this possibility, disk scans are now disabled—as are other types of scans—when taking a backup. (Bug #84422, Bug #25353234)

- **NDB Cluster APIs:** When signals were sent while the client process was receiving signals such as `SUB_GCP_COMPLETE_ACK` and `TC_COMMIT_ACK`, these signals were temporary buffered in the send buffers of the clients which sent them. If not explicitly flushed, the signals remained in these buffers until the client woke up again and flushed its buffers. Because there was no attempt made to enforce an upper limit on how long the signal could remain unsent in the local client buffers, this could lead to timeouts and other misbehavior in the components waiting for these signals.

  In addition, the fix for a previous, related issue likely made this situation worse by removing client wakeups during which the client send buffers could have been flushed.

  The current fix moves responsibility for flushing messages sent by the receivers, to the receiver (`poll_owner` client). This means that it is no longer necessary to wake up all clients merely to have them flush their buffers. Instead, the `poll_owner` client (which is already running) performs flushing the send buffer of whatever was sent while delivering signals to the recipients. (Bug #22705935)

  References: See also: Bug #18753341, Bug #23202735.

- **NDB Cluster APIs:** The adaptive send algorithm was not used as expected, resulting in every execution request being sent to the `NDB` kernel immediately, instead of trying first to collect multiple requests into larger blocks before sending them. This incurred a performance penalty on the order of 10%. The issue was due to the transporter layer always handling the *`forceSend`* argument used in several API methods (including `nextResult()` and `close()`) as `true`. (Bug #82738, Bug #24526123)

- The `ndb_print_backup_file` utility failed when attempting to read from a backup file when the backup included a table having more than 500 columns. (Bug #25302901)

  References: See also: Bug #25182956.

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

  References: See also: Bug #25302901.

- No traces were written when `ndbmtd` received a signal in any thread other than the main thread, due to the fact that all signals were blocked for other threads. This issue is fixed by the removal of `SIGBUS`, `SIGFPE`, `SIGILL`, and `SIGSEGV` signals from the list of signals being blocked. (Bug #25103068)

- The `ndb_show_tables` utility did not display type information for hash maps or fully replicated triggers. (Bug #24383742)

- The NDB Cluster Auto-Installer did not show the user how to force an exit from the application (**CTRL**+**C**). (Bug #84235, Bug #25268310)

- The NDB Cluster Auto-Installer failed to exit when it was unable to start the associated service. (Bug #84234, Bug #25268278)

- The NDB Cluster Auto-Installer failed when the port specified by the `--port` option (or the default port 8081) was already in use. Now in such cases, when the required port is not available, the next 20 ports are tested in sequence, with the first one available being used; only if all of these are in use does the Auto-Installer fail. (Bug #84233, Bug #25268221)

- Multiples instances of the NDB Cluster Auto-Installer were not detected. This could lead to inadvertent multiple deployments on the same hosts, stray processes, and similar issues. This issue is fixed by having the Auto-Installer create a PID file (`mcc.pid`), which is removed upon a successful exit. (Bug #84232, Bug #25268121)

- When a data node running with `StopOnError` set to 0 underwent an unplanned shutdown, the automatic restart performed the same type of start as the previous one. In the case where the data node had previously been started with the `--initial` option, this meant that an initial start was performed, which in cases of multiple data node failures could lead to loss of data. This issue also occurred whenever a data node shutdown led to generation of a core dump. A check is now performed to catch all such cases, and to perform a normal restart instead.

  In addition, in cases where a failed data node was unable prior to shutting down to send start phase information to the angel process, the shutdown was always treated as a startup failure, also leading to an initial restart. This issue is fixed by adding a check to execute startup failure handling only if a valid start phase was received from the client. (Bug #83510, Bug #24945638)

- Data nodes that were shut down when the redo log was exhausted did not automatically trigger a local checkpoint when restarted, and required the use of `DUMP 7099` to start one manually (see also Commands in the NDB Cluster Management Client). (Bug #82469, Bug #24412033)

- When a data node was restarted, the node was first stopped, and then, after a fixed wait, the management server assumed that the node had entered the `NOT_STARTED` state, at which point, the node was sent a start signal. If the node was not ready because it had not yet completed stopping (and was therefore not actually in `NOT_STARTED`), the signal was silently ignored.

  To fix this issue, the management server now checks to see whether the data node has in fact reached the NOT_STARTED state before sending the start signal. The wait for the node to reach this state is split into two separate checks:

  - Wait for data nodes to start shutting down (maximum 12 seconds)

  - Wait for data nodes to complete shutting down and reach `NOT_STARTED` state (maximum 120 seconds)

  If either of these cases times out, the restart is considered failed, and an appropriate error is returned. (Bug #49464, Bug #11757421)

  References: See also: Bug #28728485.

Release Series Changelogs: MySQL NDB Cluster 7.6

This section contains unified changelog information for the NDB Cluster 7.6 release series.

For changelogs covering individual MySQL NDB Cluster 7.6 releases, see NDB Cluster Release Notes.

For general information about features added in MySQL NDB Cluster 7.6, see What is New in NDB Cluster 7.6.

For an overview of features added in MySQL 5.7 that are not specific to NDB Cluster, see What Is New in MySQL 5.7. For a complete list of all bug fixes and feature changes made in MySQL 5.7 that are not specific to NDB Cluster, see the MySQL 5.7 Release Notes.

Changes in MySQL NDB Cluster 7.6.8 (5.7.24-ndb-7.6.8) (2018-10-23, General Availability)

Functionality Added or Changed

- **Performance:** This release introduces a number of significant improvements in the performance of scans; these are listed here:
  - Row checksums help detect hardware issues, but do so at the expense of performance. `NDB` now offers the possibility of disabling these by setting the new `ndb_row_checksum` server system variable to 0; doing this means that row checksums are not used for new or altered tables. This can have a significant impact (5 to 10 percent, in some cases) on performance for all types of queries. This variable is set to 1 by default, to provide compatibility with the previous behavior.
  - A query consisting of a scan can execute for a longer time in the LDM threads when the queue is not busy.
  - Previously, columns were read before checking a pushed condition; now checking of a pushed condition is done before reading any columns.
  - Performance of pushed joins should see significant improvement when using range scans as part of join execution.

Bugs Fixed

- **Packaging:** Expected NDB header files were in the `devel` RPM package instead of `libndbclient-devel`. (Bug #84580, Bug #26448330)
- **NDB Disk Data:** While restoring a local checkpoint, it is possible to insert a row that already exists in the database; this is expected behavior which is handled by deleting the existing row first, then inserting the new copy of that row. In some cases involving data on disk, `NDB` failed to delete the existing row. (Bug #91627, Bug #28341843)
- **NDB Client Programs:** Removed a memory leak in `NdbImportUtil::RangeList` that was revealed in ASAN builds. (Bug #91479, Bug #28264144)
- When copying deleted rows from a live node to a node just starting, it is possible for one or more of these rows to have a global checkpoint index equal to zero. If this happened at the same time that a full local checkpoint was started due to the undo log getting full, the `LCP_SKIP` bit was set for a row having GCI = 0, leading to an unplanned shutdown of the data node. (Bug #28372628)
- `ndbmtd` sometimes experienced a hang when exiting due to log thread shutdown. (Bug #28027150)
- When the `SUMA` kernel block receives a `SUB_STOP_REQ` signal, it executes the signal then replies with `SUB_STOP_CONF`. (After this response is relayed back to the API, the API is open to send more `SUB_STOP_REQ` signals.) After sending the `SUB_STOP_CONF`, SUMA drops the subscription if no subscribers are present, which involves sending multiple `DROP_TRIG_IMPL_REQ` messages to `DBTUP`. LocalProxy can handle up to 21 of these requests in parallel; any more than this are queued in the Short Time Queue. When execution of a `DROP_TRIG_IMPL_REQ` was delayed, there was a chance for the queue to become overloaded, leading to a data node shutdown with `Error in short time queue`.

  This issue is fixed by delaying the execution of the `SUB_STOP_REQ` signal if `DBTUP` is already handling `DROP_TRIG_IMPL_REQ` signals at full capacity, rather than queueing up the `DROP_TRIG_IMPL_REQ` signals. (Bug #26574003)
- Having a large number of deferred triggers could sometimes lead to job buffer exhaustion. This could occur due to the fact that a single trigger can execute many operations—for example, a foreign key

parent trigger may perform operations on multiple matching child table rows—and that a row operation on a base table can execute multiple triggers. In such cases, row operations are executed in batches. When execution of many triggers was deferred—meaning that all deferred triggers are executed at pre-commit—the resulting concurrent execution of a great many trigger operations could cause the data node job buffer or send buffer to be exhausted, leading to failure of the node.

This issue is fixed by limiting the number of concurrent trigger operations as well as the number of trigger fire requests outstanding per transaction.

For immediate triggers, limiting of concurrent trigger operations may increase the number of triggers waiting to be executed, exhausting the trigger record pool and resulting in the error `Too many concurrently fired triggers (increase MaxNoOfFiredTriggers`. This can be avoided by increasing `MaxNoOfFiredTriggers`, reducing the user transaction batch size, or both. (Bug #22529864)

References: See also: Bug #18229003, Bug #27310330.

- `ndbout` and `ndberr` became invalid after exiting from `mgmd_run()`, and redirecting to them before the next call to `mgmd_run()` caused a segmentation fault, during an `ndb_mgmd` service restart. This fix ensures that `ndbout` and `ndberr` remain valid at all times. (Bug #17732772, Bug #28536919)

- Running out of undo log buffer memory was reported using error 921 `Out of transaction memory ... (increase SharedGlobalMemory)`.

  This problem is fixed by introducing a new error code 923 `Out of undo buffer memory (increase UNDO_BUFFER_SIZE)`. (Bug #92125, Bug #28537319)

- When moving an `OperationRec` from the serial to the parallel queue, `Dbacc::startNext()` failed to update the `Operationrec::OP_ACC_LOCK_MODE` flag which is required to reflect the accumulated `OP_LOCK_MODE` of all previous operations in the parallel queue. This inconsistency in the ACC lock queues caused the scan lock takeover mechanism to fail, as it incorrectly concluded that a lock to take over was not held. The same failure caused an assert when aborting an operation that was a member of such an inconsistent parallel lock queue. (Bug #92100, Bug #28530928)

- A data node failed during startup due to the arrival of a `SCAN_FRAGREQ` signal during the restore phase. This signal originated from a scan begun before the node had previously failed and which should have been aborted due to the involvement of the failed node in it. (Bug #92059, Bug #28518448)

- `DBTUP` sent the error `Tuple corruption detected` when a read operation attempted to read the value of a tuple inserted within the same transaction. (Bug #92009, Bug #28500861)

- False constraint violation errors could occur when executing updates on self-referential foreign keys. (Bug #91965, Bug #28486390)

  References: See also: Bug #90644, Bug #27930382.

- An `NDB` internal trigger definition could be dropped while pending instances of the trigger remained to be executed, by attempting to look up the definition for a trigger which had already been released. This caused unpredictable and thus unsafe behavior possibly leading to data node failure. The root cause of the issue lay in an invalid assumption in the code relating to determining whether a given trigger had been released; the issue is fixed by ensuring that the behavior of `NDB`, when a trigger definition is determined to have been released, is consistent, and that it meets expectations. (Bug #91894, Bug #28451957)

- In some cases, a workload that included a high number of concurrent inserts caused data node failures when using debug builds. (Bug #91764, Bug #28384750)

- During an initial node restart with disk data tables present and `TwoPassInitialNodeRestartCopy` enabled, `DBTUP` used an unsafe scan in disk order. Such scans are no longer employed in this case. (Bug #91724, Bug #28378227)
- Checking for old LCP files tested the table version, but this was not always dependable. Now, instead of relying on the table version, the check reagrds as invalid any LCP file having a `maxGCI` smaller than its `createGci`. (Bug #91637, Bug #28346565)
- In certain cases, a cascade update trigger was fired repeatedly on the same record, which eventually consumed all available concurrent operations, leading to Error 233 `Out of operation records in transaction coordinator (increase MaxNoOfConcurrentOperations)`. If `MaxNoOfConcurrentOperations` was set to a value sufficiently high to avoid this, the issue manifested as data nodes consuming very large amounts of CPU, very likely eventually leading to a timeout. (Bug #91472, Bug #28262259)
- Inserting a row into an `NDB` table having a self-referencing foreign key that referenced a unique index on the table other than the primary key failed with `ER_NO_REFERENCED_ROW_2`. This was due to the fact that `NDB` checked foreign key constraints before the unique index was updated, so that the constraint check was unable to use the index for locating the row. Now, in such cases, `NDB` waits until all unique index values have been updated before checking foreign key constraints on the inserted row. (Bug #90644, Bug #27930382)

  References: See also: Bug #91965, Bug #28486390.
- A connection string beginning with a slash (`/`) character is now rejected by `ndb_mgmd`.

  Our thanks to Daniël van Eeden for contributing this fix. (Bug #90582, Bug #27912892)

Changes in MySQL NDB Cluster 7.6.7 (5.7.23-ndb-7.6.7) (2018-07-27, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- As part of ongoing work to improve handling of local checkpoints and minimize the occurrence of issues relating to Error 410 (`REDO log overloaded`) during LCPs, `NDB` now implements adaptive LCP control, which moderates LCP scan priority and LCP writes according to redo log usage.

  The following changes have been made with regard to `NDB` configuration parameters:

  - The default value of `RecoveryWork` is increased from 50 to 60 (60% of storage reserved for LCP files).
  - The new `InsertRecoveryWork` parameter controls the percentage of `RecoveryWork` that is reserved for insert operations. The default value is 40 (40% of `RecoveryWork`); the minimum and maximum are 0 and 70, respectively. Increasing this value allows for more writes during an LCP, while limiting the total size of the LCP. Decreasing `InsertRecoveryWork` limits the number of writes used during an LCP, but results in more space being used.

  Implementing LCP control provides several benefits to `NDB` deployments. Clusters should now survive heavy loads using default configurations much better than previously, and it should now be possible to run them reliably on systems where the available disk space is approximately 2.1 times the amount of memory allocated to the cluster (that is, the amount of `DataMemory`) or more. It is important to bear in mind that the figure just cited does not account for disk space used by tables on disk.

During load testing into a single data node with decreasing redo log sizes, it was possible to successfully load a very large quantity of data into NDB with 16GB reserved for the redo log while using no more than 50% of the redo log at any point in time.

See What is New in NDB Cluster 7.6, as well as the descriptions of the parameters mentioned previously, for more information. (Bug #90709, Bug #27942974, Bug #27942583)

References: See also: Bug #27926532, Bug #27169282.

Bugs Fixed

- **ndbinfo Information Database:** It was possible following a restart for (sometimes incomplete) fallback data to be used in populating the `ndbinfo.processes` table, which could lead to rows in this table with empty `process_name` values. Such fallback data is no longer used for this purpose. (Bug #27985339)
- An `NDB` restore operation failed under the following conditions:
  - A data node was restarted
  - The local checkpoint for the fragment being restored used two `.ctl` files
  - The first of these `.ctl` files was the file in use
  - The LCP in question consisted of more than 2002 parts

  This happened because an array used in decompression of the `.ctl` file contained only 2002 elements, which led to memory being overwritten, since this data can contain up to 2048 parts. This issue is fixed by increasing the size of the array to accommodate 2048 elements. (Bug #28303209)
- Local checkpoints did not always handle `DROP TABLE` operations correctly. (Bug #27926532)

  References: This issue is a regression of: Bug #26908347, Bug #26968613.
- During the execution of `CREATE TABLE ... IF NOT EXISTS`, the internal `open_table()` function calls `ha_ndbcluster::get_default_num_partitions()` implicitly whenever `open_table()` finds out that the requested table already exists. In certain cases, `get_default_num_partitions()` was called without the associated `thd_ndb` object being initialized, leading to failure of the statement with MySQL error 157 `Could not connect to storage engine`. Now `get_default_num_partitions()` always checks for the existence of this `thd_ndb` object, and initializes it if necessary.

Changes in MySQL NDB Cluster 7.6.6 (5.7.22-ndb-7.6.6) (2018-05-31, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- When performing an NDB backup, the `ndbinfo.logbuffers` table now displays information regarding buffer usage by the backup process on each data node. This is implemented as rows reflecting two new log types in addition to `REDO` and `DD-UNDO`. One of these rows has the log type `BACKUP-DATA`, which shows the amount of data buffer used during backup to copy fragments to backup files. The other row has the log type `BACKUP-LOG`, which displays the amount of log buffer used during the backup to record changes made after the backup has started. One each of these `log_type` rows is shown in the

`logbuffers` table for each data node in the cluster. Rows having these two log types are present in the table only while an NDB backup is currently in progress. (Bug #25822988)

- Added the `--logbuffer-size` option for `ndbd` and `ndbmtd`, for use in debugging with a large number of log messages. This controls the size of the data node log buffer; the default (32K) is intended for normal operations. (Bug #89679, Bug #27550943)

- The previously experimental shared memory (SHM) transporter is now supported in production. SHM works by transporting signals through writing them into memory, rather than on a socket. NDB already attempts to use SHM automatically between data nodes and API nodes sharing the same host. To enable explicit shared memory connections, set the `UseShm` SHM configuration parameter to `true`. When explicitly defining shared memory as the connection method, it is also necessary to identify the nodes at either end of the connection (`NodeId1` and `NodeId2` parameters), and to provide a shared memory key (`ShmKey`). In addition, to improve performance, it is also possible to set a spin time (`ShmSpinTime`) for the SHM transporter.

  Configuration of SHM is otherwise similar to that of the TCP transporter. NDB Cluster Shared-Memory Connections, provides additional information.

- The `SPJ` kernel block now takes into account when it is evaluating a join request in which at least some of the tables are used in inner joins. This means that `SPJ` can eliminate requests for rows or ranges as soon as it becomes known that a preceding request did not return any results for a parent row. This saves both the data nodes and the `SPJ` block from having to handle requests and result rows which never take part in a result row from an inner join.

  

  Note

  When upgrading from NDB 7.6.5 or earlier, you should be aware that this optimization depends on both API client and data node functionality, and so is not available until all of these have been upgraded.

- The poll receiver which `NDB` uses to read from sockets, execute messages from the sockets, and wake up other threads now offloads wakeup of other threads to a new thread that wakes up the other threads on request, and otherwise simply sleeps. This improves the scalability of a single cluster connection by keeping the receive thread from becoming overburdened by tasks including wakeup of other threads.

Bugs Fixed

- **NDB Cluster APIs:** A previous fix for an issue, in which the failure of multiple data nodes during a partial restart could cause API nodes to fail, did not properly check the validity of the associated `NdbReceiver` object before proceeding. Now in such cases an invalid object triggers handling for invalid signals, rather than a node failure. (Bug #25902137)

  References: This issue is a regression of: Bug #25092498.

- **NDB Cluster APIs:** Incorrect results, usually an empty result set, were returned when `setBound()` was used to specify a `NULL` bound. This issue appears to have been caused by a problem in gcc, limited to cases using the old version of this method (which does not employ `NdbRecord`), and is fixed by rewriting the problematic internal logic in the old implementation. (Bug #89468, Bug #27461752)

- **NDB Cluster APIs:** Released NDB API objects are kept in one or more `Ndb_free_list` structures for later reuse. Each list also keeps track of all objects seized from it, and makes sure that these are eventually released back to it. In the event that the internal function `NdbScanOperation::init()` failed, it was possible for an `NdbApiSignal` already allocated by the `NdbOperation` to be leaked. Now in such cases, `NdbScanOperation::release()` is called to release any objects allocated by the failed `NdbScanOperation` before it is returned to the free list.

This fix also handles a similar issue with `NdbOperation::init()`, where a failed call could also leak a signal. (Bug #89249, Bug #27389894)

- In some circumstances, when a transaction was aborted in the `DBTC` block, there remained links to trigger records from operation records which were not yet reference-counted, but when such an operation record was released the trigger reference count was still decremented. (Bug #27629680)

- An `NDB` online backup consists of data, which is fuzzy, and a redo and undo log. To restore to a consistent state it is necessary to ensure that the log contains all of the changes spanning the capture of the fuzzy data portion and beyond to a consistent snapshot point. This is achieved by waiting for a GCI boundary to be passed after the capture of data is complete, but before stopping change logging and recording the stop GCI in the backup's metadata.

  At restore time, the log is replayed up to the stop GCI, restoring the system to the state it had at the consistent stop GCI. A problem arose when, under load, it was possible to select a GCI boundary which occurred too early and did not span all the data captured. This could lead to inconsistencies when restoring the backup; these could be be noticed as broken constraints or corrupted `BLOB` entries.

  Now the stop GCI is chosen is so that it spans the entire duration of the fuzzy data capture process, so that the backup log always contains all data within a given stop GCI. (Bug #27497461)

  References: See also: Bug #27566346.

- For `NDB` tables, when a foreign key was added or dropped as a part of a DDL statement, the foreign key metatdata for all parent tables referenced should be reloaded in the handler on all SQL nodes connected to the cluster, but this was done only on the `mysqld` on which the statement was executed. Due to this, any subsequent queries relying on foreign key metadata from the corresponding parent tables could return inconsistent results. (Bug #27439587)

  References: See also: Bug #82989, Bug #24666177.

- `ANALYZE TABLE` used excessive amounts of CPU on large, low-cardinality tables. (Bug #27438963)

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

  References: See also: Bug #28728603.

- A data node overload could in some situations lead to an unplanned shutdown of the data node, which led to all data nodes disconnecting from the management and nodes.

  This was due to a situation in which `API_FAILREQ` was not the last received signal prior to the node failure.

  As part of this fix, the transaction coordinator's handling of `SCAN_TABREQ` signals for an `ApiConnectRecord` in an incorrect state was also improved. (Bug #27381901)

  References: See also: Bug #47039, Bug #11755287.

- In a two-node cluster, when the node having the lowest ID was started using `--nostart`, API clients could not connect, failing with `Could not alloc node id at HOST port PORT_NO: No free node id found for mysqld(API)`. (Bug #27225212)

- Changing `MaxNoOfExecutionThreads` without an initial system restart led to an unplanned data node shutdown. (Bug #27169282)

  References: This issue is a regression of: Bug #26908347, Bug #26968613.

- Race conditions sometimes occurred during asynchronous disconnection and reconnection of the transporter while other threads concurrently inserted signal data into the send buffers, leading to an unplanned shutdown of the cluster.

  As part of the work fixing this issue, the internal templating function used by the Transporter Registry when it prepares a send is refactored to use likely-or-unlikely logic to speed up execution, and to remove a number of duplicate checks for NULL. (Bug #24444908, Bug #25128512)

  References: See also: Bug #20112700.

- `ndb_restore` sometimes logged data file and log file progress values much greater than 100%. (Bug #20989106)

- The internal function `BitmaskImpl::setRange()` set one bit fewer than specified. (Bug #90648, Bug #27931995)

- It was not possible to create an `NDB` table using `PARTITION_BALANCE` set to `FOR_RA_BY_LDM_X_2`, `FOR_RA_BY_LDM_X_3`, or `FOR_RA_BY_LDM_X_4`. (Bug #89811, Bug #27602352)

  References: This issue is a regression of: Bug #81759, Bug #23544301.

- Adding a `[tcp]` or `[shm]` section to the global configuration file for a cluster with multiple data nodes caused default TCP connections to be lost to the node using the single section. (Bug #89627, Bug #27532407)

- As a result of the reuse of code intended for send threads when performing an assist send, all of the local release send buffers were released to the global pool, which caused the intended level of the local send buffer pool to be ignored. Now send threads and assisting worker threads follow their own policies for maintaining their local buffer pools. (Bug #89119, Bug #27349118)

- When sending priority A signals, we now ensure that the number of pending signals is explicitly initialized. (Bug #88986, Bug #27294856)

- In a MySQL Cluster with one MySQL Server configured to write a binary log failure occurred when creating and using an `NDB` table with non-stored generated columns. The problem arose only when the product was built with debugging support. (Bug #86084, Bug #25957586)

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

- When the internal function `ha_ndbcluster::copy_fk_for_offline_alter()` checked dependent objects on a table from which it was supposed to drop a foreign key, it did not perform any filtering for foreign keys, making it possible for it to attempt retrieval of an index or trigger instead, leading to a spurious Error 723 (`No such table`).

Changes in MySQL NDB Cluster 7.6.5 (5.7.20-ndb-7.6.5) (2018-04-20, Development)

Bugs Fixed

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

  References: See also: Bug #28698831.

- Writing of LCP control files was not always done correctly, which in some cases could lead to an unplanned shutdown of the cluster.

This fix adds the requirement that upgrades from NDB 7.6.4 (or earlier) to this release (or a later one) include initial node restarts. (Bug #26640486)

- Under certain conditions, data nodes restarted unnecessarily during execution of `ALTER TABLE... REORGANIZE PARTITION`. (Bug #25679639)

Changes in MySQL NDB Cluster 7.6.4 (5.7.20-ndb-7.6.4) (2018-01-31, Development Milestone 4)

Functionality Added or Changed

- **Incompatible Change; NDB Disk Data:** Due to changes in disk file formats, it is necessary to perform an `--initial` restart of each data node when upgrading to or downgrading from this release.
- **Important Change; NDB Disk Data:** NDB Cluster has improved node restart times and overall performance with larger data sets by implementing partial local checkpoints (LCPs). Prior to this release, an LCP always made a copy of the entire database.

  `NDB` now supports LCPs that write individual records, so it is no longer strictly necessary for an LCP to write the entire database. Since, at recovery, it remains necessary to restore the database fully, the strategy is to save one fourth of all records at each LCP, as well as to write the records that have changed since the last LCP.

  Two data node configuration parameters relating to this change are introduced in this release: `EnablePartialLcp` (default `true`, or enabled) enables partial LCPs. When partial LCPs are enabled, `RecoveryWork` controls the percentage of space given over to LCPs; it increases with the amount of work which must be performed on LCPs during restarts as opposed to that performed during normal operations. Raising this value causes LCPs during normal operations to require writing fewer records and so decreases the usual workload. Raising this value also means that restarts can take longer.

  

  Important

  Upgrading to NDB 7.6.4 or downgrading from this release requires purging then re-creating the `NDB` data node file system, which means that an initial restart of each data node is needed. An initial node restart still requires a complete LCP; a partial LCP is not used for this purpose.

  A rolling restart or system restart is a normal part of an `NDB` software upgrade. When such a restart is performed as part of an upgrade to NDB 7.6.4 or later, any existing LCP files are checked for the presence of the LCP `sysfile`, indicating that the existing data node file system was written using NDB 7.6.4 or later. If such a node file system exists, but does not contain the `sysfile`, and if any data nodes are restarted without the `--initial` option, `NDB` causes the restart to fail with an appropriate error message. This detection can be performed only as part of an upgrade; it is not possible to do so as part of a downgrade to NDB 7.6.3 or earlier from a later release.

  Exception: If there are no data node files—that is, in the event of a “clean” start or restart—using `--initial` is not required for a software upgrade, since this is already equivalent to an initial restart. (This aspect of restarts is unchanged from previous releases of NDB Cluster.)

This release also deprecates the data node configuration parameters `BackupDataBufferSize`, `BackupWriteSize`, and `BackupMaxWriteSize`; these are now subject to removal in a future NDB Cluster release. (Bug #27308632)

- **Important Change:** Added the `ndb_perror` utility for obtaining information about NDB Cluster error codes. This tool replaces `perror --ndb`; the `--ndb` option for `perror` is now deprecated and raises a warning when used; the option is subject to removal in a future NDB release.

  See `ndb_perror` — Obtain NDB Error Message Information, for more information. (Bug #81703, Bug #81704, Bug #23523869, Bug #23523926)

  References: See also: Bug #26966826, Bug #88086.

- It is now possible to specify a set of cores to be used for I/O threads performing offline multithreaded builds of ordered indexes, as opposed to normal I/O duties such as file I/O , compression , or decompression. “Offline” in this context refers to building of ordered indexes performed when the parent table is not being written to; such building takes place when an `NDB` cluster performs a node or system restart, or as part of restoring a cluster from backup using `ndb_restore --rebuild-indexes`.

  In addition, the default behaviour for offline index build work is modified to use all cores available to `ndbmtd`, rather limiting itself to the core reserved for the I/O thread. Doing so can improve restart and restore times and performance, availability, and the user experience.

  This enhancement is implemented as follows:

  1. The default value for `BuildIndexThreads` is changed from 0 to 128. This means that offline ordered index builds are now multithreaded by default.

  2. The default value for `TwoPassInitialNodeRestartCopy` is changed from `false` to `true`. This means that an initial node restart first copies all data from a “live” node to one that is starting —without creating any indexes—builds ordered indexes offline, and then again synchronizes its data with the live node, that is, synchronizing twice and building indexes offline between the two synchonizations. This causes an initial node restart to behave more like the normal restart of a node, and reduces the time required for building indexes.

  3. A new thread type (`idxbld`) is defined for the `ThreadConfig` configuration parameter, to allow locking of offline index build threads to specific CPUs.

  In addition, `NDB` now distinguishes the thread types that are accessible to “ThreadConfig” by the following two criteria:

  1. Whether the thread is an execution thread. Threads of types `main`, `ldm`, `recv`, `rep`, `tc`, and `send` are execution threads; thread types `io`, `watchdog`, and `idxbld` are not.

  2. Whether the allocation of the thread to a given task is permanent or temporary. Currently all thread types except `idxbld` are permanent.

  For additonal information, see the descriptions of the parameters in the Manual. (Bug #25835748, Bug #26928111)

- Added the `ODirectSyncFlag` configuration parameter for data nodes. When enabled, the data node treats all completed filesystem writes to the redo log as though they had been performed using `fsync`.

Note

This parameter has no effect if at least one of the following conditions is true:

- `ODirect` is not enabled.
- `InitFragmentLogFiles` is set to `SPARSE`.

(Bug #25428560)

- Added the `ndbinfo.error_messages` table, which provides information about NDB Cluster errors, including error codes, status types, brief descriptions, and classifications. This makes it possible to obtain error information using SQL in the `mysql` client (or other MySQL client program), like this:

```
mysql> SELECT * FROM ndbinfo.error_messages WHERE error_code='321';
+------------+---------------------+-----------------+----------------------+
| error_code | error_description   | error_status    | error_classification |
+------------+---------------------+-----------------+----------------------+
|        321 | Invalid nodegroup id | Permanent error | Application error    |
+------------+---------------------+-----------------+----------------------+
1 row in set (0.00 sec)
```

  The query just shown provides equivalent information to that obtained by issuing `ndb_perror 321` or (now deprecated) `perror --ndb 321` on the command line. (Bug #86295, Bug #26048272)

- `ThreadConfig` now has an additional `nosend` parameter that can be used to prevent a `main`, `ldm`, `rep`, or `tc` thread from assisting the send threads, by setting this parameter to 1 for the given thread. By default, `nosend` is 0. It cannot be used with threads other than those of the types just listed.

- When executing a scan as a pushed join, all instances of `DBSPJ` were involved in the execution of a single query; some of these received multiple requests from the same query. This situation is improved by enabling a single SPJ request to handle a set of root fragments to be scanned, such that only a single SPJ request is sent to each `DBSPJ` instance on each node and batch sizes are allocated per fragment, the multi-fragment scan can obtain a larger total batch size, allowing for some scheduling optimizations to be done within `DBSPJ`, which can scan a single fragment at a time (giving it the total batch size allocation), scan all fragments in parallel using smaller sub-batches, or some combination of the two.

  Since the effect of this change is generally to require fewer SPJ requests and instances, performance of pushed-down joins should be improved in many cases.

- As part of work ongoing to optimize bulk DDL performance by ndbmtd, it is now possible to obtain performance improvements by increasing the batch size for the bulk data parts of DDL operations which process all of the data in a fragment or set of fragments using a scan. Batch sizes are now made configurable for unique index builds, foreign key builds, and online reorganization, by setting the respective data node configuration parameters listed here:

  - `MaxFKBuildBatchSize`: Maximum scan batch size used for building foreign keys.
  - `MaxReorgBuildBatchSize`: Maximum scan batch size used for reorganization of table partitions.
  - `MaxUIBuildBatchSize`: Maximum scan batch size used for building unique keys.

  For each of the parameters just listed, the default value is 64, the minimum is 16, and the maximum is 512.

Increasing the appropriate batch size or sizes can help amortize inter-thread and inter-node latencies and make use of more parallel resources (local and remote) to help scale DDL performance.

- Formerly, the data node `LGMAN` kernel block processed undo log records serially; now this is done in parallel. The `rep` thread, which hands off undo records to local data handler (LDM) threads, waited for an LDM to finish applying a record before fetching the next one; now the `rep` thread no longer waits, but proceeds immediately to the next record and LDM.

  There are no user-visible changes in functionality directly associated with this work; this performance enhancement is part of the work being done in NDB 7.6 to improve undo long handling for partial local checkpoints.

- When applying an undo log the table ID and fragment ID are obtained from the page ID. This was done by reading the page from `PGMAN` using an extra `PGMAN` worker thread, but when applying the undo log it was necessary to read the page again.

  This became very inefficient when using `O_DIRECT` (see `ODirect`) since the page was not cached in the OS kernel.

  Mapping from page ID to table ID and fragment ID is now done using information the extent header contains about the table IDs and fragment IDs of the pages used in a given extent. Since the extent pages are always present in the page cache, no extra disk reads are required to perform the mapping, and the information can be read using existing `TSMAN` data structures.

- Added the `NODELOG DEBUG` command in the `ndb_mgm` client to provide runtime control over data node debug logging. `NODE DEBUG ON` causes a data node to write extra debugging information to its node log, the same as if the node had been started with `--verbose`. `NODELOG DEBUG OFF` disables the extra logging.

- Added the `LocationDomainId` configuration parameter for management, data, and API nodes. When using NDB Cluster in a cloud environment, you can set this parameter to assign a node to a given availability domain or availability zone. This can improve performance in the following ways:

  - If requested data is not found on the same node, reads can be directed to another node in the same availability domain.
  - Communication between nodes in different availability domains are guaranteed to use `NDB` transporters' WAN support without any further manual intervention.
  - The transporter's group number can be based on which availability domain is used, such that also SQL and other API nodes communicate with local data nodes in the same availability domain whenever possible.
  - The arbitrator can be selected from an availability domain in which no data nodes are present, or, if no such availability domain can be found, from a third availability domain.

  This parameter takes an integer value between 0 and 16, with 0 being the default; using 0 is the same as leaving `LocationDomainId` unset.

Bugs Fixed

- **Important Change:** The `--passwd` option for `ndb_top` is now deprecated. It is removed (and replaced with `--password`) in NDB 7.6.5. (Bug #88236, Bug #20733646)

  References: See also: Bug #86615, Bug #26236320, Bug #26907833.

- **NDB Disk Data:** An `ALTER TABLE` that switched the table storage format between `MEMORY` and `DISK` was always performed in place for all columns. This is not correct in the case of a column whose storage format is inherited from the table; the column's storage type is not changed.

  For example, this statement creates a table `t1` whose column `c2` uses in-memory storage since the table does so implicitly:

  ```
  CREATE TABLE t1 (c1 INT PRIMARY KEY, c2 INT) ENGINE NDB;
  ```

  The `ALTER TABLE` statement shown here is expected to cause `c2` to be stored on disk, but failed to do so:

  ```
  ALTER TABLE t1 STORAGE DISK TABLESPACE ts1;
  ```

  Similarly, an on-disk column that inherited its storage format from the table to which it belonged did not have the format changed by `ALTER TABLE ... STORAGE MEMORY`.

  These two cases are now performed as a copying alter, and the storage format of the affected column is now changed. (Bug #26764270)

- Errors in parsing `NDB_TABLE` modifiers could cause memory leaks. (Bug #26724559)

- Added `DUMP` code 7027 to facilitate testing of issues relating to local checkpoints. For more information, see DUMP 7027. (Bug #26661468)

- A previous fix intended to improve logging of node failure handling in the transaction coordinator included logging of transactions that could occur in normal operation, which made the resulting logs needlessly verbose. Such normal transactions are no longer written to the log in such cases. (Bug #26568782)

  References: This issue is a regression of: Bug #26364729.

- Due to a configuration file error, CPU locking capability was not available on builds for Linux platforms. (Bug #26378589)

- Some `DUMP` codes used for the `LGMAN` kernel block were incorrectly assigned numbers in the range used for codes belonging to `DBTUX`. These have now been assigned symbolic constants and numbers in the proper range (10001, 10002, and 10003). (Bug #26365433)

- Node failure handling in the `DBTC` kernel block consists of a number of tasks which execute concurrently, and all of which must complete before TC node failure handling is complete. This fix extends logging coverage to record when each task completes, and which tasks remain, includes the following improvements:

  - Handling interactions between GCP and node failure handling interactions, in which TC takeover causes GCP participant stall at the master TC to allow it to extend the current GCI with any transactions that were taken over; the stall can begin and end in different GCP protocol states. Logging coverage is extended to cover all scenarios. Debug logging is now more consistent and understandable to users.

  - Logging done by the `QMGR` block as it monitors duration of node failure handling duration is done more frequently. A warning log is now generated every 30 seconds (instead of 1 minute), and this now includes `DBDIH` block debug information (formerly this was written separately, and less often).

  - To reduce space used, `DBTC instance number:` is shortened to `DBTC number:`.

- A new error code is added to assist testing.

(Bug #26364729)

- During a restart, `DBLQH` loads redo log part metadata for each redo log part it manages, from one or more redo log files. Since each file has a limited capacity for metadata, the number of files which must be consulted depends on the size of the redo log part. These files are opened, read, and closed sequentially, but the closing of one file occurs concurrently with the opening of the next.

  In cases where closing of the file was slow, it was possible for more than 4 files per redo log part to be open concurrently; since these files were opened using the `OM_WRITE_BUFFER` option, more than 4 chunks of write buffer were allocated per part in such cases. The write buffer pool is not unlimited; if all redo log parts were in a similar state, the pool was exhausted, causing the data node to shut down.

  This issue is resolved by avoiding the use of `OM_WRITE_BUFFER` during metadata reload, so that any transient opening of more than 4 redo log files per log file part no longer leads to failure of the data node. (Bug #25965370)

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

- A join entirely within the materialized part of a semi-join was not pushed even if it could have been. In addition, `EXPLAIN` provided no information about why the join was not pushed. (Bug #88224, Bug #27022925)

  References: See also: Bug #27067538.

- When the duplicate weedout algorithm was used for evaluating a semi-join, the result had missing rows. (Bug #88117, Bug #26984919)

  References: See also: Bug #87992, Bug #26926666.

- A table used in a loose scan could be used as a child in a pushed join query, leading to possibly incorrect results. (Bug #87992, Bug #26926666)

- When representing a materialized semi-join in the query plan, the MySQL Optimizer inserted extra `QEP_TAB` and `JOIN_TAB` objects to represent access to the materialized subquery result. The join pushdown analyzer did not properly set up its internal data structures for these, leaving them uninitialized instead. This meant that later usage of any item objects referencing the materialized semi-join accessed an initialized `tableno` column when accessing a 64-bit `tableno` bitmask, possibly referring to a point beyond its end, leading to an unplanned shutdown of the SQL node. (Bug #87971, Bug #26919289)

- In some cases, a `SCAN_FRAGCONF` signal was received after a `SCAN_FRAGREQ` with a close flag had already been sent, clearing the timer. When this occurred, the next `SCAN_FRAGREF` to arrive caused time tracking to fail. Now in such cases, a check for a cleared timer is performed prior to processing the `SCAN_FRAGREF` message. (Bug #87942, Bug #26908347)

- While deleting an element in `Dbacc`, or moving it during hash table expansion or reduction, the method used (`getLastAndRemove()`) could return a reference to a removed element on a released page, which could later be referenced from the functions calling it. This was due to a change brought about by the implementation of dynamic index memory in NDB 7.6.2; previously, the page had always belonged to a single `Dbacc` instance, so accessing it was safe. This was no longer the case following the change; a page released in `Dbacc` could be placed directly into the global page pool where any other thread could then allocate it.

Now we make sure that newly released pages in `Dbacc` are kept within the current `Dbacc` instance and not given over directly to the global page pool. In addition, the reference to a released page has been removed; the affected internal method now returns the last element by value, rather than by reference. (Bug #87932, Bug #26906640)

References: See also: Bug #87987, Bug #26925595.

- The `DBTC` kernel block could receive a `TCRELEASEREQ` signal in a state for which it was unprepared. Now it such cases it responds with a `TCRELEASECONF` message, and subsequently behaves just as if the API connection had failed. (Bug #87838, Bug #26847666)

  References: See also: Bug #20981491.

- When a data node was configured for locking threads to CPUs, it failed during startup with `Failed to lock tid`.

  This was is a side effect of a fix for a previous issue, which disabled CPU locking based on the version of the available `glibc`. The specific `glibc` issue being guarded against is encountered only in response to an internal NDB API call (`Ndb_UnlockCPU()`) not used by data nodes (and which can be accessed only through internal API calls). The current fix enables CPU locking for data nodes and disables it only for the relevant API calls when an affected `glibc` version is used. (Bug #87683, Bug #26758939)

  References: This issue is a regression of: Bug #86892, Bug #26378589.

- `ndb_top` failed to build on platforms where the `ncurses` library did not define `stdscr`. Now these platforms require the `tinfo` library to be included. (Bug #87185, Bug #26524441)

- On completion of a local checkpoint, every node sends a `LCP_COMPLETE_REP` signal to every other node in the cluster; a node does not consider the LCP complete until it has been notified that all other nodes have sent this signal. Due to a minor flaw in the LCP protocol, if this message was delayed from another node other than the master, it was possible to start the next LCP before one or more nodes had completed the one ongoing; this caused problems with `LCP_COMPLETE_REP` signals from previous LCPs becoming mixed up with such signals from the current LCP, which in turn led to node failures.

  To fix this problem, we now ensure that the previous LCP is complete before responding to any `TCGETOPSIZEREQ` signal initiating a new LCP. (Bug #87184, Bug #26524096)

- NDB Cluster did not compile successfully when the build used `WITH_UNIT_TESTS=OFF`. (Bug #86881, Bug #26375985)

- Recent improvements in local checkpoint handling that use `OM_CREATE` to open files did not work correctly on Windows platforms, where the system tried to create a new file and failed if it already existed. (Bug #86776, Bug #26321303)

- A potential hundredfold signal fan-out when sending a `START_FRAG_REQ` signal could lead to a node failure due to a `job buffer full` error in start phase 5 while trying to perform a local checkpoint during a restart. (Bug #86675, Bug #26263397)

  References: See also: Bug #26288247, Bug #26279522.

- Compilation of NDB Cluster failed when using `-DWITHOUT_SERVER=1` to build only the client libraries. (Bug #85524, Bug #25741111)

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

Changes in MySQL NDB Cluster 7.6.3 (5.7.18-ndb-7.6.3) (2017-07-03, Development Milestone 3)

Functionality Added or Changed

- In some critical situations such as data node failure, it was possible for the volume of log messages produced to cause file system and other issues, which compounded the problem, due to the fact that these messages were logged synchronously using `stdout`. To keep this from happening, log messages from worker threads now use a log buffer instead, which is nonblocking, and thus much less likely to cause interference with other processes under such conditions. (Bug #24748843)

- Added the `--diff-default` option for `ndb_config`. This option causes the program to print only those parameters having values that differ from their defaults. (Bug #85831, Bug #25844166)

- Added the `ndb_top` program on unix-based platforms. This utility shows CPU load and usage information for an `NDB` data node, with periodic updates (each second, by default). The display is in text or color ASCII graph format; both formats can be displayed at the same time. It is also possible to disable color output for the graph.

  `ndb_top` connects to an NDB Cluster SQL node—that is, a MySQL Server—and for this reason must be able to connect as a MySQL user having the `SELECT` privilege on tables in the `ndbinfo` database.

  `ndb_top` is not currently available for Windows platforms.

  For more information, see `ndb_top` — View CPU usage information for NDB threads.

Bugs Fixed

- **Packaging:** Two missing dependencies were added to the `apt` packages:

  - The data node package requires `libclass-methodmaker-perl`
  - The auto-installer requires `python-paramiko`

  (Bug #85679, Bug #25799465)

- **NDB Disk Data:** If the tablespace for a disk table had been fully consumed when a node failed, and table rows were deleted and inserted—or updated with shrinking or expanding disk column values—while the node was unavailable, a subsequent restart could fail with error 1601 `Out of extents, tablespace full`. We prevent this from happening by reserving 4 percent of the tablespace for use during node starts. (Bug #25923125)

- **NDB Cluster APIs:** The implementation method `NdbDictionary::NdbTableImpl::getColumn()`, used from many places in the NDB API where a column is referenced by name, has been made more efficient. This method used a linear search of an array of columns to find the correct column object, which could be inefficient for tables with many columns, and was detected as a significant use of CPU in customer applications. (Ideally, users should perform name-to-column object mapping, and then use column IDs or objects in method calls, but in practice this is not always done.) A less costly hash index implementation, used previously for the name lookup, is reinstated for tables having relatively many columns. (A linear search continues to be used for tables having fewer columns, where the difference in performance is neglible.) (Bug #24829435)

- **NDB Cluster APIs:** NDB error 631 is reclassified as the (temporary) node recovery error `Scan take over error, restart scan transaction`. This was previously exposed to applications as an internal (and permanent) error which provided no description. (Bug #86401, Bug #26116231)

- Backup `.log` files contained log entries for one or more extra fragments, due to an issue with filtering out changes logged by other nodes in the same node group. This resulted in a larger `.log` file and thus use of more resources than necessary; it could also cause problems when restoring, since backups from different nodes could interfere with one another while the log was being applied. (Bug #25891014)

- Memory exhaustion during fragment creation led to an unplanned shutdown of the cluster. This issue could be triggered by the addition of unique keys to a large number of columns at the same time. (Bug #25851801)

- When making the final write to a redo log file, it is expected that the next log file is already opened for writes, but this was not always the case with a slow disk, leading to node failure. Now in such cases `NDB` waits for the next file to be opened properly before attempting to write to it. (Bug #25806659)

- Data node threads can be bound to a single CPU or a set of CPUs, a set of CPUs being represented internally by `NDB` as a `SparseBitmask`. When attempting to lock to a set of CPUs, CPU usage was excessive due to the fact that the routine performing the locks used the `mt_thr_config.cpp::do_bind()` method, which looks for bits that are set over the entire theoretical range of the `SparseBitmask` (2^{32}-2, or 4294967294). This is fixed by using `SparseBitmask::getBitNo()`, which can be used to iterate over only those bits that are actually set, instead. (Bug #25799506)

- Setting `NoOfFragmentLogParts` such that there were more than 4 redo log parts per local data manager led to resource exhaustion and subsequent multiple data node failures. Since this is an invalid configuration, a check has been added to detect a configuration with more than 4 redo log parts per LDM, and reject it as invalid. (Bug #25333414)

- In certain cases, a failed `ALTER TABLE ... ADD UNIQUE KEY` statement could lead to SQL node failure. (Bug #24444878)

  References: This issue is a regression of: Bug #23089566.

- Error 240 is raised when there is a mismatch between foreign key trigger columns and the values supplied to them during trigger execution, but had no error message indicating the source of the problem. (Bug #23141739)

  References: See also: Bug #23068914, Bug #85857.

- If the number of LDM blocks was not evenly divisible by the number of TC/SPJ blocks, SPJ requests were not equally distributed over the available SPJ instances. Now a round-robin distribution is used to distribute SPJ requests across all available SPJ instances more effectively.

  As part of this work, a number of unused member variables have been removed from the class `Dbtc`. (Bug #22627519)

- `ALTER TABLE .. MAX_ROWS=0` can now be performed only by using a copying `ALTER TABLE` statement. Resetting `MAX_ROWS` to 0 can no longer be performed using `ALGORITHM=INPLACE`. (Bug #21960004)

- During a system restart, when a node failed due to having missed sending heartbeats, all other nodes reported only that another node had failed without any additional information. Now in such cases, the fact that heartbeats were missed and the ID of the node that failed to send heartbeats is reported in both the error log and the data node log. (Bug #21576576)

- Due to a previous issue with unclear separation between the optimize and execute phases when a query involved a `GROUP BY`, the join-pushable evaluator was not sure whether its optimized query execution plan was in fact pushable. For this reason, such grouped joins were always considered not pushable. It has been determined that the separation issue has been resolved by work already done in MySQL 5.6, and so we now remove this limitation. (Bug #86623, Bug #26239591)

- When deleting all rows from a table immediately followed by `DROP TABLE`, it was possible that the shrinking of the `DBACC` hash index was not ready prior to the drop. This shrinking is a per-fragment operation that does not check the state of the table. When a table is dropped, `DBACC` releases resources, during which the description of the fragment size and page directory is not consistent; this could lead to reads of stale pages, and undefined behavior.

  Inserting a great many rows followed by dropping the table should also have had such effects due to expansion of the hash index.

  To fix this problem we make sure, when a fragment is about to be released, that there are no pending expansion or shrinkage operations on this fragment. (Bug #86449, Bug #26138592)

- Some error messages still referred to `IndexMemory`, although that parameter has been deprecated. (Bug #86385, Bug #26107514)

- The internal function `execute_signals()` in `mt.cpp` read three section pointers from the signal even when none was passed to it. This was mostly harmless, although unneeded. When the signal read was the last one on the last page in the job buffer, and the next page in memory was not mapped or otherwise accessible, `ndbmtd` failed with an error. To keep this from occurring, this function now only reads section pointers that are actually passed to it. (Bug #86354, Bug #26092639)

- There was at most one attempt in Dbacc to remove hash index pages freed when a table was dropped. This meant that, for large partitions (32 pages or more) there were always some pages lost. Now all hash index pages are freed when table using them is dropped. (Bug #86247, Bug #26030894)

- When a query on an `NDB` table failed due to a foreign key constraint violation, no useful information about the foreign key was shown in the error message, which contained only the text `Unknown error code`. (Bug #86241, Bug #26029485, Bug #16371292)

  References: See also: Bug #16275684.

- The `ndb_show_tables` program `--unqualified` option did not work correctly when set to 0 (false); this should disable the option and so cause fully qualified table and index names to be printed in the output. (Bug #86017, Bug #25923164)

- When an `NDB` table with foreign key constraints is created, its indexes are created first, and then, during foreign key creation, these indexes are loaded into the `NDB` dictionary cache. When a `CREATE TABLE` statement failed due to an issue relating to foreign keys, the indexes already in the cache were not invalidated. This meant that any subsequent `CREATE TABLE` with any indexes having the same names as those in the failed statement produced inconsistent results. Now, in such cases, any indexes named in the failed `CREATE TABLE` are immediately invalidated from the cache. (Bug #85917, Bug #25882950)

- During a local checkpoint, the record size is obtained from the `DBTUP` kernel block. This record size remained in use until the LCP scan was completed, which made it possible for `DBTUP` to update the maximum record size on commit of an `ALTER TABLE` that added a column to the table, and which could lead to node failure during the LCP. Now the record size is fetched at a point where updating it does not lead to this condition. (Bug #85858, Bug #25860002)

- Attempting to execute `ALTER TABLE ... ADD FOREIGN KEY` when the key to be added had the name of an existing foreign key on the same table failed with the wrong error message. (Bug #85857, Bug #23068914)

- The node internal scheduler (in `mt.cpp`) collects statistics about its own progress and any outstanding work it is performing. One such statistic is the number of outstanding send bytes, collected in `send_buffer::m_node_total_send_buffer_size`. This information may later be used by the send thread scheduler, which uses it as a metric to tune its own send performance versus latency.

  In order to reduce lock contention on the internal send buffers, they are split into two `thr_send_buffer` parts, `m_buffer` and `m_sending`, each protected by its own mutex, and their combined size repesented by `m_node_total_send_buffer_size`.

  Investigation of the code revealed that there was no consistency as to which mutex was used to update `m_node_total_send_buffer_size`, with the result that there was no consurrency protection for this value. To avoid this, `m_node_total_send_buffer_size` is replaced with two values, `m_buffered_size` and `m_sending_size`, which keep separate track of the sizes of the two buffers. These counters are updated under the protection of two different mutexes protecting each buffer individually, and are now added together to obtain the total size.

  With concurrency control established, updates of the partial counts should now be correct, so that their combined value no longer accumulates errors over time. (Bug #85687, Bug #25800933)

- Enabled the use of short or packed short `TRANSID_AI` signals for sending results from `DBSPJ` back to the client API. (Bug #85545, Bug #25750355)

  References: See also: Bug #85525, Bug #25741170.

- The maximum `BatchByteSize` as sent in `SCANREQ` signals was not always set correctly to reflect a limited byte size available in the client result buffers. The result buffer size calculation has been modified such that the effective batch byte size accurately reflects the maximum that may be returned by data nodes to prevent a possible overflow of the result buffers. (Bug #85411, Bug #25703113)

- When compiling the NDB kernel with `gcc` version 6.0.0 or later, it is now built using `-flifetime-dse=1`. (Bug #85381, Bug #25690926)

Changes in MySQL NDB Cluster 7.6.2 (5.7.18-ndb-7.6.2) (2017-04-26, Development Milestone 2)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Incompatible Change; NDB Disk Data:** Due to changes in disk file formats, it is necessary to perform an `--initial` restart of each data node when upgrading to or downgrading from this release.

- **Important Change:** As part of an ongoing effort to simplify NDB Cluster configuration, memory for indexes is now allocated dynamically from `DataMemory`; the `IndexMemory` configuration parameter is now deprecated, and is subject to removal in a future release. Any memory which has been set for `IndexMemory` in the `config.ini` file is now automatically added to `DataMemory`. In addition, the default value for `DataMemory` has been increased to 98M, and the default for `IndexMemory` has been decreased to 0.

  In addition to simplifying configuration of `NDB`, a further benefit of these changes is that scaling up by increasing the number of LDM threads is no longer limited by having set an insufficiently large value for `IndexMemory`. Previously, it was sometimes the case that increasing the number of LDM threads could lead to index memory exhaustion while large amounts of `DataMemory` remained available.

Because instances of the DBACC kernel block (responsible for hash index storage) now share memory with each one another as well as with DBLQH (the kernel block that acts as the local data manager), they can take advantage of the fact that scaling up does not increase DataMemory usage greatly, and make use of spare memory for indexes freely. (For more information about these kernel blocks, see The DBACC Block, and The DBLQH Block.) In other words, index memory is no longer a static quantity allocated to each DBACC instance only once, on startup of the cluster, but rather this resource can now be allocated and deallocated whenever conditions require it.

Related changes which have been made as part of this work are listed here:

- Several instances of DataMemory usage not related to storage of table data now use transaction memory instead.

  For this reason, it may be necessary on some systems to increase SharedGlobalMemory. In addition, systems performing initial bulk loads of data using large transactions may need to break up large transactions into smaller ones.

- Data nodes now generate MemoryUsage events (see NDB Cluster Log Events) and write appropriate messages in the cluster log when resource usage reaches 99%, in addition to when it reaches 80%, 90%, or 100% as they did previously.

- REPORT MEMORYUSAGE and other commands which expose memory consumption now shows index memory consumption using a page size of 32K rather than 8K.

- IndexMemory is no longer one of the values displayed in the ndbinfo.memoryusage table's memory_type column.

- The ndbinfo.resources table now shows the DISK_OPERATIONS resource as TRANSACTION_MEMORY.

  The RESERVED resource has been removed.

- IndexMemory is no longer displayed in ndb_config output.

- **Performance:** A number of debugging statements and printouts in the sources for the DBTC and DBLQH kernel blocks, as well as in related code, were moved into debugging code or removed altogether. This is expected to result in an improvement of up to 10% in the performance of local data management and transaction coordinator threads in many common use cases.

- **NDB Cluster APIs; ndbinfo Information Database:** Added two tables to the ndbinfo information database. The config_nodes table provides information about nodes that are configured as part of a given NDB Cluster, such as node ID and process type. The processes table shows information about nodes currently connected to the cluster; this information includes the process name and system process ID, and service address. For each data node and SQL node, it also shows the process ID of the node's angel process.

  As part of the work done to implement the processes table, a new set_service_uri() method has been added to the NDB API.

  For more information, see The ndbinfo config_nodes Table, and The ndbinfo processes Table, as well as Ndb_cluster_connection::set_service_uri().

- **NDB Cluster APIs:** The system name of an NDB cluster is now visible in the mysql client as the value of the Ndb_system_name status variable, and can also be obtained by NDB API application using the Ndb_cluster_connection::get_system_name() method. The system name can be set using the Name parameter in the [system] section of the cluster configuration file.

- Added the `--query-all` option to `ndb_config`. This option acts much like the `--query` option except that `--query-all` (short form: `-a`) dumps configuration information for all attributes at one time. (Bug #60095, Bug #11766869)

- Previously, when one LDM thread experienced I/O lag, such as during a disk overload condition, it wrote to a local checkpoint more slowly—that is, it wrote in I/O lag mode. However, other LDM threads did not necessarily observe or conform to this state. To ensure that write speed for the LCP is reduced by all LDM threads when such a slowdown is encountered, `NDB` now tracks I/O lag mode globally, so that I/O lag state is reported as soon as at least one thread is writing in I/O lag mode, and thus all LDM threads are forced to write in lag mode while the lag condition persists. This reduction in write speed by other LDM instances should increase overall capacity, enabling the disk overload condition to be overcome more quickly in such cases than before.

- Added the `ndb_import` tool to facilitate the loading of CSV-formatted data, such as that produced by `SELECT INTO OUTFILE`, into an `NDB` table. `ndb_import` is intended to function much like `mysqlimport` or the `LOAD DATA INFILE` SQL statement, and supports many similar options for formatting of the data file. A connection to an `NDB` management server (`ndb_mgmd`) is required; there must be an otherwise unused `[api]` slot in the cluster's `config.ini` file for this purpose. In addition, the target database and table (created using the `NDB` storage engine) must already exist, and the name of the CSV file (less any file extension) must be the same as that of the target table. A running SQL node is needed for creating the target database and table, but is not required for `ndb_import` to function.

  For more information, see `ndb_import` — Import CSV Data Into NDB.

Bugs Fixed

- **Partitioning:** The output of `EXPLAIN PARTITIONS` displayed incorrect values in the `partitions` column when run on an explicitly partitioned `NDB` table having a large number of partitions.

  This was due to the fact that, when processing an `EXPLAIN` statement, `mysqld` calculates the partition ID for a hash value as (*`hash_value % number_of_partitions`*), which is correct only when the table is partitioned by `HASH`, since other partitioning types use different methods of mapping hash values to partition IDs. This fix replaces the partition ID calculation performed by `mysqld` with an internal `NDB` function which calculates the partition ID correctly, based on the table's partitioning type. (Bug #21068548)

  References: See also: Bug #25501895, Bug #14672885.

- **Microsoft Windows:** When collecting information about CPUs on Windows, the Auto-Installer counted only physical cores, unlike on other platforms, where it collects information about both physical and virtual cores. Now the CPU information obtained on Windows is the same as that provided on other platforms. (Bug #85209, Bug #25636423)

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

- When `ndb_report_thresh_binlog_epoch_slip` was enabled, an event buffer status message with `report_reason=LOW/ENOUGH_FREE_EVENTBUFFER` was printed in the logs when event buffer usage was high and then decreased to a lower level. This calculation was based on total allocated event buffer memory rather than the limit set by `ndb_eventbuffer_max_alloc`; it was also printed even when the event buffer had unlimited memory (`ndb_eventbuffer_max_alloc` = 0, the default), which could confuse users.

  This is fixed as follows:

- The calculation of `ndb_eventbuffer_free_percent` is now based on `ndb_eventbuffer_max_alloc`, rather than the amount actually allocated.
- When `ndb_eventbuffer_free_percent` is set and `ndb_eventbuffer_max_alloc` is equal to 0, event buffer status messages using `report_reason=LOW/ENOUGH_FREE_EVENTBUFFER` are no longer printed.
- When `ndb_report_thresh_binlog_epoch_slip` is set, an event buffer status message showing `report_reason=BUFFERED_EPOCHS_OVER_THRESHOLD` is written each 10 seconds (rather than every second) whenever this is greater than the threshold.

(Bug #25726723)

- A bulk update is executed by reading records and executing a transaction on the set of records, which is started while reading them. When transaction initialization failed, the transaction executor function was subsequently unaware that this had occurred, leading to SQL node failures. This issue is fixed by providing appropriate error handling when attempting to initialize the transaction. (Bug #25476474)

  References: See also: Bug #20092754.

- CPU usage of the data node's main thread by the `DBDIH` master block as the end of a local checkpoint could approach 100% in certain cases where the database had a very large number of fragment replicas. This is fixed by reducing the frequency and range of fragment queue checking during an LCP. (Bug #25443080)

- Execution of an online `ALTER TABLE ... REORGANIZE PARTITION` statement on an `NDB` table having a primary key whose length was greater than 80 bytes led to restarting of data nodes, causing the reorganization to fail. (Bug #25152165)

- Multiple data node failures during a partial restart of the cluster could cause API nodes to fail. This was due to expansion of an internal object ID map by one thread, thus changing its location in memory, while another thread was still accessing the old location, leading to a segmentation fault in the latter thread.

  The internal `map()` and `unmap()` functions in which this issue arose have now been made thread-safe. (Bug #25092498)

  References: See also: Bug #25306089.

- The planned shutdown of an NDB Cluster having more than 10 data nodes was not always performed gracefully. (Bug #20607730)

- Dropped `TRANS_AI` signals that used the long signal format were not handled by the `DBTC` kernel block. (Bug #85606, Bug #25777337)

  References: See also: Bug #85519, Bug #27540805.

- Iproved pushed join handling by eliminating unneeded `FLUSH_AI` attributes that passed an empty row to the `DBSPJ` kernel block, when a row should be passed to the SPJ API only; this reduces the set of `AttrInfo` projections that must be executed in order to produce the result. This also makes it possible to employ packed `TRANSID_AI` signals when delivering SPJ API results, which is more efficient. (Bug #85525, Bug #25741170)

  References: See also: Bug #85545, Bug #25750355.

- Use of the long signal format (introduced in NDB 6.4) for an incoming `TRANSID_AI` message is supported by the `BACKUP`, `DBTC`, `DBLQH`, `SUMA`, `DBSPJ`, and `DBUTIL` NDB kernel blocks, but the `DBTUP`

block produced long signals only when sending to DPSPJ or DBUTIL, and otherwise sent a series of short signals instead. Now DBTUP uses long signals for such messages whenever the receiving block supports this optimization. (Bug #85519, Bug #25740805)

- To prevent a scan from returning more rows, bytes, or both than the client has reserved buffers for, the DBTUP kernel block reports the size of the TRANSID_AI it has sent to the client in the TUPKEYCONF signal it sends to the requesting DBLQH block. DBLQH is aware of the maximum batch size available for the result set, and terminates the scan batch if this has been exceeded.

  The DBSPJ block's FLUSH_AI attribute allows DBTUP to produce two TRANSID_AI results from the same row, one for the client, and one for DBSPJ, which is needed for key lookups on the joined tables. The size of both of these were added to the read length reported by the DBTUP block, which caused the controlling DBLQH block to believe that it had consumed more of the available maximum batch size than was actually the case, leading to premature termination of the scan batch which could have a negative impact on performance of SPJ scans. To correct this, only the actual read length part of an API request is now reported in such cases. (Bug #85408, Bug #25702850)

- Data node binaries for Solaris 11 built using Oracle Developer Studio 12.5 on SPARC platforms failed with bus errors. (Bug #85390, Bug #25695818)

- During the initial phase of a scan request, the DBTC kernel block sends a series of DIGETNODESREQ signals to the DBDIH block in order to obtain dictionary information for each fragment to be scanned. If DBDIH returned DIGETNODESREF, the error code from that signal was not read, and Error 218 Out of LongMessageBuffer was always returned instead. Now in such cases, the error code from the DIGETNODESREF signal is actually used. (Bug #85225, Bug #25642405)

- If the user attempts to invoke ndb_setup.py while the Auto-Installer is still running—for example, after closing the terminal in which it was started and later opening a new terminal and invoking it in the new one—the program fails with the error Web server already running, which is expected behavior. In such cases, the mcc.pid file must first be removed prior to restarting the Auto-Installer (also expected behavior). Now when the program fails for this reason, the location of mcc.pid is included in the error message to simplify this task. (Bug #85169, Bug #25611093)

- The planned shutdown of a data node after one or more data nodes in the same node group had failed was not always performed correctly. (Bug #85168, Bug #25610703)

- There existed the possibility of a race condition between schema operations on the same database object originating from different SQL nodes; this could occur when one of the SQL nodes was late in releasing its metadata lock on the affected schema object or objects in such a fashion as to appear to the schema distribution coordinator that the lock release was acknowledged for the wrong schema change. This could result in incorrect application of the schema changes on some or all of the SQL nodes or a timeout with repeated waiting max ### sec for distributing... messages in the node logs due to failure of the distribution protocol. (Bug #85010, Bug #25557263)

  References: See also: Bug #24926009.

- When a foreign key was added to or dropped from an NDB table using an ALTER TABLE statement, the parent table's metadata was not updated, which made it possible to execute invalid alter operations on the parent afterwards.

  Until you can upgrade to this release, you can work around this problem by running SHOW CREATE TABLE on the parent immediately after adding or dropping the foreign key; this statement causes the table's metadata to be reloaded. (Bug #82989, Bug #24666177)

- Transactions on NDB tables with cascading foreign keys returned inconsistent results when the query cache was also enabled, due to the fact that mysqld was not aware of child table updates. This meant

that results for a later `SELECT` from the child table were fetched from the query cache, which at that point contained stale data.

This is fixed in such cases by adding all children of the parent table to an internal list to be checked by `NDB` for updates whenever the parent is updated, so that `mysqld` is now properly informed of any updated child tables that should be invalidated from the query cache. (Bug #81776, Bug #23553507)

Changes in MySQL NDB Cluster 7.6.1 (5.7.17-ndb-7.6.1) (Not released, Development Milestone 1)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **NDB Disk Data:** A new file format is introduced in this release for NDB Disk Data tables. The new format provides a mechanism whereby each Disk Data table can be uniquely identified without reusing table IDs. This is intended to help resolve issues with page and extent handling that were visible to the user as problems with rapid creating and dropping of Disk Data tables, and for which the old format did not provide a ready means to fix.

  The new format is now used whenever new undo log file groups and tablespace data files are created. Files relating to existing Disk Data tables continue to use the old format until their tablespaces and undo log file groups are re-created. *Important*: The old and new formats are not compatible and so cannot be employed for different data files or undo log files that are used by the same Disk Data table or tablespace.

  To avoid problems relating to the old format, you should re-create any existing tablespaces and undo log file groups when upgrading. You can do this by performing an initial restart of each data node (that is, using the `--initial` option) as part of the upgrade process. Since the current release is a pre-GA Developer release, this initial node restart is optional for now, but *you should expect it—and prepare for it now—to be mandatory in GA versions of NDB 7.6*.

  If you are using Disk Data tables, a downgrade from *any* NDB 7.6 release to any NDB 7.5 or earlier release requires restarting data nodes with `--initial` as part of the downgrade process, due to the fact that NDB 7.5 and earlier releases cannot read the new Disk Data file format.

  For more information, see Upgrading and Downgrading NDB Cluster.

Bugs Fixed

- **Packaging:** NDB Cluster Auto-Installer RPM packages for SLES 12 failed due to a dependency on `python2-crypto` instead of `python-pycrypto`. (Bug #25399608)
- **NDB Disk Data:** Stale data from NDB Disk Data tables that had been dropped could potentially be included in backups due to the fact that disk scans were enabled for these. To prevent this possibility, disk scans are now disabled—as are other types of scans—when taking a backup. (Bug #84422, Bug #25353234)
- **NDB Cluster APIs:** When signals were sent while the client process was receiving signals such as `SUB_GCP_COMPLETE_ACK` and `TC_COMMIT_ACK`, these signals were temporary buffered in the send buffers of the clients which sent them. If not explicitly flushed, the signals remained in these buffers until the client woke up again and flushed its buffers. Because there was no attempt made to enforce an upper limit on how long the signal could remain unsent in the local client buffers, this could lead to timeouts and other misbehavior in the components waiting for these signals.

In addition, the fix for a previous, related issue likely made this situation worse by removing client wakeups during which the client send buffers could have been flushed.

The current fix moves responsibility for flushing messages sent by the receivers, to the receiver (`poll_owner` client). This means that it is no longer necessary to wake up all clients merely to have them flush their buffers. Instead, the `poll_owner` client (which is already running) performs flushing the send buffer of whatever was sent while delivering signals to the recipients. (Bug #22705935)

References: See also: Bug #18753341, Bug #23202735.

- **NDB Cluster APIs:** The adaptive send algorithm was not used as expected, resulting in every execution request being sent to the `NDB` kernel immediately, instead of trying first to collect multiple requests into larger blocks before sending them. This incurred a performance penalty on the order of 10%. The issue was due to the transporter layer always handling the *`forceSend`* argument used in several API methods (including `nextResult()` and `close()`) as `true`. (Bug #82738, Bug #24526123)

- The `ndb_print_backup_file` utility failed when attempting to read from a backup file when the backup included a table having more than 500 columns. (Bug #25302901)

  References: See also: Bug #25182956.

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

  References: See also: Bug #25302901.

- No traces were written when `ndbmtd` received a signal in any thread other than the main thread, due to the fact that all signals were blocked for other threads. This issue is fixed by the removal of `SIGBUS`, `SIGFPE`, `SIGILL`, and `SIGSEGV` signals from the list of signals being blocked. (Bug #25103068)

- The `ndb_show_tables` utility did not display type information for hash maps or fully replicated triggers. (Bug #24383742)

- The NDB Cluster Auto-Installer did not show the user how to force an exit from the application (**CTRL**+**C**). (Bug #84235, Bug #25268310)

- The NDB Cluster Auto-Installer failed to exit when it was unable to start the associated service. (Bug #84234, Bug #25268278)

- The NDB Cluster Auto-Installer failed when the port specified by the `--port` option (or the default port 8081) was already in use. Now in such cases, when the required port is not available, the next 20 ports are tested in sequence, with the first one available being used; only if all of these are in use does the Auto-Installer fail. (Bug #84233, Bug #25268221)

- Multiples instances of the NDB Cluster Auto-Installer were not detected. This could lead to inadvertent multiple deployments on the same hosts, stray processes, and similar issues. This issue is fixed by having the Auto-Installer create a PID file (`mcc.pid`), which is removed upon a successful exit. (Bug #84232, Bug #25268121)

- When a data node running with `StopOnError` set to 0 underwent an unplanned shutdown, the automatic restart performed the same type of start as the previous one. In the case where the data node had previously been started with the `--initial` option, this meant that an initial start was performed, which in cases of multiple data node failures could lead to loss of data. This issue also occurred whenever a data node shutdown led to generation of a core dump. A check is now performed to catch all such cases, and to perform a normal restart instead.

In addition, in cases where a failed data node was unable prior to shutting down to send start phase information to the angel process, the shutdown was always treated as a startup failure, also leading to an initial restart. This issue is fixed by adding a check to execute startup failure handling only if a valid start phase was received from the client. (Bug #83510, Bug #24945638)

- Data nodes that were shut down when the redo log was exhausted did not automatically trigger a local checkpoint when restarted, and required the use of `DUMP 7099` to start one manually (see also Commands in the NDB Cluster Management Client). (Bug #82469, Bug #24412033)

- When a data node was restarted, the node was first stopped, and then, after a fixed wait, the management server assumed that the node had entered the `NOT_STARTED` state, at which point, the node was sent a start signal. If the node was not ready because it had not yet completed stopping (and was therefore not actually in `NOT_STARTED`), the signal was silently ignored.

  To fix this issue, the management server now checks to see whether the data node has in fact reached the NOT_STARTED state before sending the start signal. The wait for the node to reach this state is split into two separate checks:

  - Wait for data nodes to start shutting down (maximum 12 seconds)

  - Wait for data nodes to complete shutting down and reach `NOT_STARTED` state (maximum 120 seconds)

  If either of these cases times out, the restart is considered failed, and an appropriate error is returned. (Bug #49464, Bug #11757421)

  References: See also: Bug #28728485.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

W