# MySQL NDB Cluster 7.4 Release Notes

**Abstract**

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

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

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

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

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

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

For legal information, see the Legal Notices.

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

Document generated on: 2022-10-21 (revision: 25564)

# Table of Contents

# Preface and Legal Notices

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

## Legal Notices

## Documentation Accessibility

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

## Access to Oracle Support for Accessibility

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

# Changes in MySQL NDB Cluster 7.4.38 (5.6.51-ndb-7.4.38) (2022-10-12, General Availability)

MySQL NDB Cluster 7.4.38 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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

Version 5.6.51-ndb-7.4.38 has no release notes, or they have not been published because the product version has not been released.

# Changes in MySQL NDB Cluster 7.4.37 (5.6.51-ndb-7.4.37) (2022-07-27, General Availability)

MySQL NDB Cluster 7.4.37 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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

- Functionality Added or Changed
- Bugs Fixed

## Functionality Added or Changed

- **Important Change:** The linked OpenSSL library for MySQL Server has been updated to version 1.1.1o. Issues fixed in OpenSSL version 1.1.1o are described at https://www.openssl.org/news/cl111.txt and https://www.openssl.org/news/vulnerabilities.html. (Bug #34133985)

## Bugs Fixed

- Path lengths were not always calculated correctly by the data nodes. (Bug #33993607)
- Some `NDB` internal signals were not always checked properly. (Bug #33896428)
- If a `CR_UNKNOWN_ERROR` was to be sent to a client, an exception could occur. (Bug #31933415)

# Changes in MySQL NDB Cluster 7.4.36 (5.6.51-ndb-7.4.36) (2022-04-27, General Availability)

MySQL NDB Cluster 7.4.36 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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

## Bugs Fixed

- In some cases, `NDB` did not validate all node IDs of data nodes correctly. (Bug #33896409)
- In some cases, array indexes were not handled correctly. (Bug #33896389, Bug #33896399, Bug #33916134)

# Changes in MySQL NDB Cluster 7.4.35 (5.6.51-ndb-7.4.35) (2022-01-19, General Availability)

MySQL NDB Cluster 7.4.35 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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

## Bugs Fixed

- **NDB Cluster APIs:** It is no longer possible to use the `DIVERIFYREQ` signal asynchronously. (Bug #33161562)
- Added missing values checks in `ndbd` and `ndbmtd`. (Bug #33661024)
- In certain cases, an event's category was not properly detected. (Bug #33304814)
- `DBDICT` did not always perform table name checks correctly. (Bug #33161548)
- `SET_LOGLEVELORD` signals were not always handled correctly. (Bug #33161246)
- `DUMP 11001` did not always handle all of its arguments correctly. (Bug #33157513)
- File names were not always verified correctly. (Bug #33157475)
- Added a number of missing ID and other values checks in `ndbd` and `ndbmtd`. (Bug #32983700, Bug #32893708, Bug #32957478, Bug #32983256, Bug #32983339, Bug #32983489, Bug #32983517, Bug #33157527, Bug #33157531, Bug #33161271, Bug #33161298, Bug #33161314, Bug #33161331, Bug #33161372, Bug #33161462, Bug #33161511, Bug #33161519, Bug #33161537, Bug #33161570, Bug #33162059, Bug #33162065, Bug #33162074, Bug #33162082, Bug #33162092, Bug #33162098, Bug #33304819)
- The management server did not always handle events of the wrong size correctly. (Bug #32957547)

# Changes in MySQL NDB Cluster 7.4.34 (5.6.51-ndb-7.4.34) (2021-10-20, General Availability)

MySQL NDB Cluster 7.4.34 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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

## Bugs Fixed

- NDB Cluster could not be compiled using GCC 10 or 11. (Bug #33282549)
- A buffer used in the `SUMA` kernel block did not always accommodate multiple signals. (Bug #33246047)

- It was possible in certain cases for an array index to exceed `NO_OF_BUCKETS`. (Bug #33019959)
- Added an `ndbrequire()` in `QMGR` to check whether the node ID received from the `CM_REGREF` signal is less than `MAX_NDB_NODES`. (Bug #32983311)
- A check was reported missing from the code for handling `GET_TABLEID_REQ` signals. To fix this issue, all code relating to all `GET_TABLEID_*` signals has been removed from the `NDB` sources, since these signals are no longer used or supported in NDB Cluster. (Bug #32983249)
- It was possible in some cases to specify an invalid node type when working with the internal management API. Now the API specifically disallows invalid node types, and defines an “unknown” node type (`NDB_MGM_NODE_TYPE_UNKNOWN`) to cover such cases. (Bug #32957364)
- `ndb_restore` raised a warning to use `--disable-indexes` when restoring data after the metadata had already been restored with `--disable-indexes`.

  When `--disable-indexes` is used to restore metadata before restoring data, the tables in the target schema have no indexes. We now check when restoring data with this option to ensure that there are no indexes on the target table, and print the warning only if the table already has indexes. (Bug #28749799)
- When restoring of metadata was done using `--disable-indexes`, there was no attempt to create indexes or foreign keys dependent on these indexes, but when `ndb_restore` was used without the option, indexes and foreign keys were created. When `--disable-indexes` was used later while restoring data, `NDB` attempted to drop any indexes created in the previous step, but ignored the failure of a drop index operation due to a dependency on the index of a foreign key which had not been dropped. This led subsequently to problems while rebuilding indexes, when there was an attempt to create foreign keys which already existed.

  We fix `ndb_restore` as follows:

  - When `--disable-indexes` is used, `ndb_restore` now drops any foreign keys restored from the backup.
  - `ndb_restore` now checks for the existence of indexes before attempting to drop them.

  (Bug #26974491)

# Changes in MySQL NDB Cluster 7.4.33 (5.6.51-ndb-7.4.33) (2021-07-21, General Availability)

MySQL NDB Cluster 7.4.33 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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

## Bugs Fixed

- **Packaging:** The `ndb-common` man page was removed, and the information it contained moved to other man pages. (Bug #32799519)
- `Ndb_rep_tab_key` member variables were not null-terminated before being logged. (Bug #32841430)

  References: See also: Bug #32393245.

# Changes in MySQL NDB Cluster 7.4.32 (5.6.51-ndb-7.4.32) (2021-04-21, General Availability)

MySQL NDB Cluster 7.4.32 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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

# Changes in MySQL NDB Cluster 7.4.31 (5.6.51-ndb-7.4.31) (2021-01-19, General Availability)

MySQL NDB Cluster 7.4.31 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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

- Deprecation and Removal Notes
- Bugs Fixed

## Deprecation and Removal Notes

- **NDB Client Programs:** Effective with this release, the MySQL NDB Cluster Auto-Installer (`ndb_setup.py`) has been has been removed from the NDB Cluster binary and source distributions, and is no longer supported. (Bug #32084831)

  References: See also: Bug #31888835.

- **ndbmemcache:** `ndbmemcache`, which was deprecated in the previous release of NDB Cluster, has now been removed from NDB Cluster, and is no longer supported. (Bug #32106576)

## Bugs Fixed

- Using the maximum size of an index key supported by index statistics (3056 bytes) caused buffer issues in data nodes. (Bug #32094904)

  References: See also: Bug #25038373.

- When a table creation schema transaction is prepared, the table is in `TS_CREATING` state, and is changed to `TS_ACTIVE` state when the schema transaction commits on the `DBDIH` block. In the case where the node acting as `DBDIH` coordinator fails while the schema transaction is committing, another node starts taking over for the coordinator. The following actions are taken when handling this node failure:

  - `DBDICT` rolls the table creation schema transaction forward and commits, resulting in the table involved changing to `TS_ACTIVE` state.

  - `DBDIH` starts removing the failed node from tables by moving active table replicas on the failed node from a list of stored fragment replicas to another list.

These actions are performed asynchronously many times, and when interleaving may cause a race condition. As a result, the replica list in which the replica of a failed node resides becomes nondeterministic and may differ between the recovering node (that is, the new coordinator) and other `DIH` participant nodes. This difference violated a requirement for knowing which list the failed node's replicas can be found during the recovery of the failed node recovery on the other participants.

To fix this, moving active table replicas now covers not only tables in `TS_ACTIVE` state, but those in `TS_CREATING` (prepared) state as well, since the prepared schema transaction is always rolled forward.

In addition, the state of a table creation schema transaction which is being aborted is now changed from `TS_CREATING` or `TS_IDLE` to `TS_DROPPING`, to avoid any race condition there. (Bug #30521812)

# Changes in MySQL NDB Cluster 7.4.30 (5.6.50-ndb-7.4.30) (2020-10-20, General Availability)

MySQL NDB Cluster 7.4.30 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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 5.6 through MySQL 5.6.50 (see Changes in MySQL 5.6.50 (2020-10-19, General Availability)).

- Deprecation and Removal Notes
- Bugs Fixed

## Deprecation and Removal Notes

- **NDB Cluster APIs:** Support for Node.js has been removed in this release.

  Node.js continues to be supported in NDB Cluster 8.0 only. (Bug #31781948)

- **NDB Client Programs:** Effective with this release, the MySQL NDB Cluster Auto-Installer (`ndb_setup.py`) has been deprecated and is subject to removal in a future version of NDB Cluster. (Bug #31888835)

- **ndbmemcache:** `ndbmemcache` is deprecated in this release of NDB Cluster, and is scheduled for removal in the next release. (Bug #31876970)

## Bugs Fixed

- **Packaging:** The Dojo library included with NDB Cluster has been upgraded to version 1.15.4. (Bug #31559518)

# Changes in MySQL NDB Cluster 7.4.29 (5.6.49-ndb-7.4.29) (2020-07-14, General Availability)

MySQL NDB Cluster 7.4.29 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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 5.6 through MySQL 5.6.49 (see Changes in MySQL 5.6.49 (2020-07-13, General Availability)).

## Bugs Fixed

- During a node restart, the `SUMA` block of the node that is starting must get a copy of the subscriptions (events with subscribers) and subscribers (`NdbEventOperation` instances which are executing) from a node already running. Before the copy is complete, nodes which are still starting ignore any user-level `SUB_START` or `SUB_STOP` requests; after the copy is done, they can participate in such requests. While the copy operation is in progress, user-level `SUB_START` and `SUB_STOP` requests are blocked using a `DICT` lock.

  An issue was found whereby a starting node could participate in `SUB_START` and `SUB_STOP` requests after the lock was requested, but before it is granted, which resulted in unsuccessful `SUB_START` and `SUB_STOP` requests. This fix ensures that the nodes cannot participate in these requests until after the `DICT` lock has actually been granted. (Bug #31302657)

- The Dojo toolkit included with NDB Cluster and used by the Auto-Installer was upgraded to version 1.15.3. (Bug #31029110)

- A packed version 1 configuration file returned by `ndb_mgmd` could contain duplicate entries following an upgrade to NDB 8.0, which made the file incompatible with clients using version 1. This occurs due to the fact that the code for handling backwards compatibility assumed that the entries in each section were already sorted when merging it with the default section. To fix this, we now make sure that this sort is performed prior to merging. (Bug #31020183)

- When executing any of the `SHUTDOWN`, `ALL STOP`, or `ALL RESTART` management commands, it is possible for different nodes to attempt to stop on different global checkpoint index (CGI) boundaries. If they succeed in doing so, then a subsequent system restart is slower than normal because any nodes having an earlier stop GCI must undergo takeover as part of the process. When nodes failing on the first GCI boundary cause surviving nodes to be nonviable, surviving nodes suffer an arbitration failure; this has the positive effect of causing such nodes to halt at the correct GCI, but can give rise to spurious errors or similar.

  To avoid such issues, extra synchronization is now performed during a planned shutdown to reduce the likelihood that different data nodes attempt to shut down at different GCIs as well as the use of unnecessary node takeovers during system restarts. (Bug #31008713)

# Changes in MySQL NDB Cluster 7.4.28 (5.6.48-ndb-7.4.28) (2020-04-28, General Availability)

MySQL NDB Cluster 7.4.28 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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 5.6 through MySQL 5.6.48 (see Changes in MySQL 5.6.48 (2020-04-27, General Availability)) .

## Functionality Added or Changed

- **NDB Client Programs:** Removed a dependency from the `ndb_waiter` and `ndb_show_tables` utility programs on the `NDBT` library. This library, used in `NDB` development for testing, is not required for normal use. The visible effect for users from this change is that these programs no longer print `NDBT_ProgramExit - status` following completion of a run. Applications that depend upon this behavior should be updated to reflect this change when upgrading to this release.

- Added the `--ndb-log-fail-terminate` option for `mysqld`. When used, this causes the SQL node to terminate if it is unable to log all row events. (Bug #21911930)

  References: See also: Bug #30383919.

## Bugs Fixed

- When a node ID allocation request failed with **`NotMaster`** temporary errors, the node ID allocation was always retried immediately, without regard to the cause of the error. This caused a very high rate of retries, whose effects could be observed as an excessive number of `Alloc node id for node nnn failed` log messages (on the order of 15,000 messages per second). (Bug #30293495)

- For `NDB` tables having no explicit primary key, `NdbReceiverBuffer` could be allocated with too small a size. This was due to the fact that the attribute bitmap sent to `NDB` from the data nodes always includes the primary key. The extra space required for hidden primary keys is now taken into consideration in such cases. (Bug #30183466)

# Changes in MySQL NDB Cluster 7.4.27 (5.6.47-ndb-7.4.27) (2020-01-14, General Availability)

MySQL NDB Cluster 7.4.27 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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 5.6 through MySQL 5.6.47 (see Changes in MySQL 5.6.47 (2020-01-13, General Availability)) .

## Bugs Fixed

- If a transaction was aborted while getting a page from the disk page buffer and the disk system was overloaded, the transaction hung indefinitely. This could also cause restarts to hang and node failure handling to fail. (Bug #30397083, Bug #30360681)

  References: See also: Bug #30152258.

- The maximum global checkpoint (GCP) commit lag and GCP save timeout are recalculated whenever a node shuts down, to take into account the change in number of data nodes. This could lead to the unintentional shutdown of a viable node when the threshold decreased below the previous value. (Bug #27664092)

  References: See also: Bug #26364729.

- Concurrent `SELECT` and `ALTER TABLE` statements on the same SQL node could sometimes block one another while waiting for locks to be released. (Bug #17812505, Bug #30383887)

# Changes in MySQL NDB Cluster 7.4.26 (5.6.46-ndb-7.4.26) (2019-10-15, General Availability)

MySQL NDB Cluster 7.4.26 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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 5.6 through MySQL 5.6.46 (see Changes in MySQL 5.6.46 (2019-10-14, General Availability)).

## Bugs Fixed

- During a restart when the data nodes had started but not yet elected a president, the management server received a `node ID already in use` error, which resulted in excessive retries and logging. This is fixed by introducing a new error 1705 `Not ready for connection allocation yet` for this case.

  During a restart when the data nodes had not yet completed node failure handling, a spurious `Failed to allocate nodeID` error was returned. This is fixed by adding a check to detect an incomplete node start and to return error 1703 `Node failure handling not completed` instead.

  As part of this fix, the frequency of retries has been reduced for `not ready to alloc nodeID` errors, an error insert has been added to simulate a slow restart for testing purposes, and log messages have been reworded to indicate that the relevant node ID allocation errors are minor and only temporary. (Bug #27484514)

# Changes in MySQL NDB Cluster 7.4.25 (5.6.45-ndb-7.4.25) (2019-07-23, General Availability)

MySQL NDB Cluster 7.4.25 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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 5.6 through MySQL 5.6.45 (see Changes in MySQL 5.6.45 (2019-07-22, General Availability)).

## Bugs Fixed

- The `requestInfo` fields for the long and short forms of the `LQHKEYREQ` signal had different definitions; bits used for the key length in the short version were reused for flags in the long version, since the key length is implicit in the section length of the long version of the signal but it was possible for long `LQHKEYREQ` signals to contain a keylength in these same bits, which could be

misinterpreted by the receiving local query handler, potentially leading to errors. Checks have now been implemented to make sure that this no longer happens. (Bug #29820838)

- When restoring `TINYBLOB` columns, `ndb_restore` now treats them as having the `BINARY` character set. (Bug #29486538)
- Restoration of epochs by `ndb_restore` failed due to temporary redo errors. Now `ndb_restore` retries epoch updates when such errors occur. (Bug #29466089)
- `ndb_restore --restore-epoch` incorrectly reported the stop GCP as 1 less than the actual position. (Bug #29343655)
- Added support which was missing in `ndb_restore` for conversions between the following sets of types:
  - `BLOB` and `BINARY` or `VARBINARY` columns
  - `TEXT` and `BLOB` columns
  - `BLOB` columns with unequal lengths
  - `BINARY` and `VARBINARY` columns with unequal lengths

  (Bug #28074988)
- Restore points in backups created with the `SNAPSHOTSTART` option (see Using The NDB Cluster Management Client to Create a Backup) were not always consistent with epoch boundaries. (Bug #27566346)

  References: See also: Bug #27497461.

# Changes in MySQL NDB Cluster 7.4.24 (5.6.44-ndb-7.4.24) (2019-04-26, General Availability)

MySQL NDB Cluster 7.4.24 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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 5.6 through MySQL 5.6.44 (see Changes in MySQL 5.6.44 (2019-04-25, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

## Functionality Added or Changed

- Building with `CMake3` is now supported by the `compile-cluster` script included in the `NDB` source distribution.

## Bugs Fixed

- **Important Change:** The dependency of `ndb_restore` on the `NDBT` library, which is used for internal testing only, has been removed. This means that the program no longer prints

`NDBT_ProgramExit: ...` when terminating. Applications that depend upon this behavior should be updated to reflect this change when upgrading to this release.

- When a pushed join executing in the `DBSPJ` block had to store correlation IDs during query execution, memory for these was allocated for the lifetime of the entire query execution, even though these specific correlation IDs are required only when producing the most recent batch in the result set. Subsequent batches require additional correlation IDs to be stored and allocated; thus, if the query took sufficiently long to complete, this led to exhaustion of query memory (error 20008). Now in such cases, memory is allocated only for the lifetime of the current result batch, and is freed and made available for re-use following completion of the batch. (Bug #29336777)

  References: See also: Bug #26995027.

- In some cases, one and sometimes more data nodes underwent an unplanned shutdown while running `ndb_restore`. This occurred most often, but was not always restircted to, when restoring to a cluster having a different number of data nodes from the cluster on which the original backup had been taken.

  The root cause of this issue was exhaustion of the pool of `SafeCounter` objects, used by the `DBDICT` kernel block as part of executing schema transactions, and taken from a per-block-instance pool shared with protocols used for `NDB` event setup and subscription processing. The concurrency of event setup and subscription processing is such that the `SafeCounter` pool can be exhausted; event and subscription processing can handle pool exhaustion, but schema transaction processing could not, which could result in the node shutdown experienced during restoration.

  This problem is solved by giving `DBDICT` schema transactions an isolated pool of reserved `SafeCounters` which cannot be exhausted by concurrent `NDB` event activity. (Bug #28595915)

- `ndb_restore` did not restore autoincrement values correctly when one or more staging tables were in use. As part of this fix, we also in such cases block applying of the `SYSTAB_0` backup log, whose content continued to be applied directly based on the table ID, which could ovewrite the autoincrement values stored in `SYSTAB_0` for unrelated tables. (Bug #27917769, Bug #27831990)

  References: See also: Bug #27832033.

- `ndb_restore` employed a mechanism for restoring autoincrement values which was not atomic, and thus could yield incorrect autoincrement values being restored when multiple instances of `ndb_restore` were used in parallel. (Bug #27832033)

  References: See also: Bug #27917769, Bug #27831990.

- When executing the redo log in debug mode it was possible for a data node to fail when deallocating a row. (Bug #93273, Bug #28955797)

- An `NDB` table having both a foreign key on another `NDB` table using `ON DELETE CASCADE` and one or more `TEXT` or `BLOB` columns leaked memory.

  As part of this fix, `ON DELETE CASCADE` is no longer supported for foreign keys on `NDB` tables when the child table contains a column that uses any of the `BLOB` or `TEXT` types. (Bug #89511, Bug #27484882)

# Changes in MySQL NDB Cluster 7.4.23 (5.6.43-ndb-7.4.23) (2019-01-22, General Availability)

MySQL NDB Cluster 7.4.23 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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 5.6 through MySQL 5.6.43 (see Changes in MySQL 5.6.43 (2019-01-21, General Availability)).

## Bugs Fixed

- **NDB Disk Data:** When a log file group had more than 18 undo logs, it was not possible to restart the cluster. (Bug #251155785)

  References: See also: Bug #28922609.

- **NDB Replication:** When writes on the master—done in such a way that multiple changes affecting `BLOB` column values belonging to the same primary key were part of the same epoch—were replicated to the slave, Error 1022 occurred due to constraint violations in the `NDB$BLOB_id_part` table. (Bug #28746560)

- When a local checkpoint (LCP) was complete on all data nodes except one, and this node failed, `NDB` did not continue with the steps required to finish the LCP. This led to the following issues:

  No new LCPs could be started.

  Redo and Undo logs were not trimmed and so grew excessively large, causing an increase in times for recovery from disk. This led to write service failure, which eventually led to cluster shutdown when the head of the redo log met the tail. This placed a limit on cluster uptime.

  Node restarts were no longer possible, due to the fact that a data node restart requires that the node's state be made durable on disk before it can provide redundancy when joining the cluster. For a cluster with two data nodes and two fragment replicas, this meant that a restart of the entire cluster (system restart) was required to fix the issue (this was not necessary for a cluster with two fragment replicas and four or more data nodes). (Bug #28728485, Bug #28698831)

  References: See also: Bug #11757421.

- It was possible in certain cases for nodes to hang during an initial restart. (Bug #28698831)

  References: See also: Bug #27622643.

- When tables with `BLOB` columns were dropped and then re-created with a different number of `BLOB` columns the event definitions for monitoring table changes could become inconsistent in certain error situations involving communication errors when the expected cleanup of the corresponding events was not performed. In particular, when the new versions of the tables had more `BLOB` columns than the original tables, some events could be missing. (Bug #27072756)

- When running a cluster with 4 or more data nodes under very high loads, data nodes could sometimes fail with Error 899 `Rowid already allocated`. (Bug #25960230)

- When starting, a data node copies metadata, while a local checkpoint updates metadata. To avoid any conflict, any ongoing LCP activity is paused while metadata is being copied. An issue arose when a local checkpoint was paused on a given node, and another node that was also restarting checked for a complete LCP on this node; the check actually caused the LCP to be completed before copying of metadata was complete and so ended the pause prematurely. Now in such cases, the LCP completion check waits to complete a paused LCP until copying of metadata is finished and the pause ends as expected, within the LCP in which it began. (Bug #24827685)

- Asynchronous disconnection of `mysqld` from the cluster caused any subsequent attempt to start an NDB API transaction to fail. If this occurred during a bulk delete operation, the SQL layer called `HA::end_bulk_delete()`, whose implementation by `ha_ndbcluster` assumed that a transaction had been started, and could fail if this was not the case. This problem is fixed by checking that the transaction pointer used by this method is set before referencing it. (Bug #20116393)

# Changes in MySQL NDB Cluster 7.4.22 (5.6.42-ndb-7.4.22) (2018-10-23, General Availability)

MySQL NDB Cluster 7.4.22 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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 5.6 through MySQL 5.6.42 (see Changes in MySQL 5.6.42 (2018-10-22, General Availability)).

## Bugs Fixed

- **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 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.

# Changes in MySQL NDB Cluster 7.4.21 (5.6.41-ndb-7.4.21) (2018-07-27, General Availability)

MySQL NDB Cluster 7.4.21 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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 5.6 through MySQL 5.6.41 (see Changes in MySQL 5.6.41 (2018-07-27, General Availability)).

## Bugs Fixed

- **NDB Cluster APIs:** When `Ndb::dropEventOperation()` tried to clean up a pending event, it failed to clear a pointer to the list of GCI operations being deleted and discarded (`Gci_ops` object), so that this pointer referred to a deleted object. GCI operations arriving after this could then be inserted as part of the next such list belonging to the now-deleted object, leading to memory corruption and other issues. (Bug #90011, Bug #27675005)

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

  References: See also: Bug #28698831.

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

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

# Changes in MySQL NDB Cluster 7.4.20 (5.6.40-ndb-7.4.20) (2018-04-20, General Availability)

MySQL NDB Cluster 7.4.20 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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 5.6 through MySQL 5.6.40 (see Changes in MySQL 5.6.40 (2018-04-19, General Availability)).

## Bugs Fixed

- **NDB Cluster APIs:** The maximum time to wait which can be specified when calling either of the NDB API methods `Ndb::pollEvents()` or `pollEvents2()` was miscalculated such that the method could wait up to 9 ms too long before returning to the client. (Bug #88924, Bug #27266086)

- Under certain conditions, data nodes restarted unnecessarily during execution of `ALTER TABLE... REORGANIZE PARTITION`. (Bug #25675481)

  References: See also: Bug #26735618, Bug #27191468.

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

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

  References: See also: Bug #20112700.

# Changes in MySQL NDB Cluster 7.4.19 (5.6.39-ndb-7.4.19) (2018-01-23, General Availability)

MySQL NDB Cluster 7.4.19 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

NDB 7.4.19 replaces the NDB 7.4.18 release, and is the successor to NDB 7.4.17. Users of NDB 7.4.17 and previous NDB 7.4 releases should upgrade directly to MySQL NDB Cluster 7.4.19 or newer.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases (including the NDB 7.4.18 release which this release replaces), as well as all bug fixes and feature changes which were added in mainline MySQL 5.6 through MySQL 5.6.39 (see Changes in MySQL 5.6.39 (2018-01-15, General Availability)).

## Bugs Fixed

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

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

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

- A query against the `INFORMATION_SCHEMA.FILES` table returned no results when it included an `ORDER BY` clause. (Bug #26877788)

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

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

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

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

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

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

- When representing a materialized semijoin 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 semijoin 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)

- 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 7.4.18 (5.6.39-ndb-7.4.18) (2018-01-17, General Availability)

MySQL NDB Cluster 7.4.18 was replaced following release by NDB 7.4.19. Users of NDB 7.4.17 and previous NDB 7.4 releases should upgrade directly to MySQL NDB Cluster 7.4.19 or later.

For changes that originally appeared in NDB 7.4.18, see Changes in MySQL NDB Cluster 7.4.19 (5.6.39-ndb-7.4.19) (2018-01-23, General Availability).

# Changes in MySQL NDB Cluster 7.4.17 (5.6.38-ndb-7.4.17) (2017-10-18, General Availability)

MySQL NDB Cluster 7.4.17 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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 5.6 through MySQL 5.6.38 (see Changes in MySQL 5.6.38 (2017-10-16, General Availability)).

## Bugs Fixed

- 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.

- 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)

- 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.

# Changes in MySQL NDB Cluster 7.4.16 (5.6.37-ndb-7.4.16) (2017-07-18, General Availability)

MySQL NDB Cluster 7.4.16 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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 5.6 through MySQL 5.6.37 (see Changes in MySQL 5.6.37 (2017-07-17, General Availability)).

## Functionality Added or Changed

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

## Bugs Fixed

- **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)
- **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)
- 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)

- 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.

- 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)

- 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)

- 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` or the `ONLINE` keyword. (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)

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

- 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)

- 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)

- 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)

- 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)

- 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.

- 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)

- 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 7.4.15 (5.6.36-ndb-7.4.15) (2017-04-10, General Availability)

MySQL NDB Cluster 7.4.15 is a new release of MySQL NDB Cluster 7.4, based on MySQL Server 5.6 and including features in version 7.4 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

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 5.6 through MySQL 5.6.36 (see Changes in MySQL 5.6.36 (2017-04-10, General Availability)).

## 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.

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

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

- 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)

- 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.

- 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.

- 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 7.4.14 (5.6.35-ndb-7.4.14) (2017-01-17, General Availability)

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

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

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

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 5.6 through MySQL 5.6.35 (see Changes in MySQL 5.6.35 (2016-12-12, General Availability)).

## Bugs Fixed

- `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.

- Queries against the `ndbinfo.memory_per_fragment` table when running with a large number of data nodes could produce unexpected results for the highest-numbered nodes. (Bug #25176404)

- The `rand()` function was used to produce a unique table ID and table version needed to identify a schema operation distributed between multiple SQL nodes, relying on the assumption that `rand()` would never produce the same numbers on two different instances of mysqld. It was later determined that this is not the case, and that in fact it is very likely for the same random numbers to be produced on all SQL nodes.

  This fix removes the usage of `rand()` for producing a unique table ID or version, and instead uses a sequence in combination with the node ID of the coordinator. This guarantees uniqueness until the counter for the sequence wraps, which should be sufficient for this purpose.

The effects of this duplication could be observed as timeouts in the log (for example `NDB create db: waiting max 119 sec for distributing`) when restarting multiple `mysqld` processes simultaneously or nearly so, or when issuing the same `CREATE DATABASE` or `DROP DATABASE` statement on multiple SQL nodes. (Bug #24926009)

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

- Long message buffer exhaustion when firing immediate triggers could result in row ID leaks; this could later result in persistent `RowId already allocated` errors (`NDB` Error 899). (Bug #23723110)

  References: See also: Bug #19506859, Bug #13927679.

- when a parent `NDB` table in a foreign key relationship was updated, the update cascaded to a child table as expected, but the change was not cascaded to a child table of this child table (that is, to a grandchild of the original parent). This can be illustrated using the tables generated by the following `CREATE TABLE` statements:

  ```
  CREATE TABLE parent(
    id INT PRIMARY KEY AUTO_INCREMENT,
    col1 INT UNIQUE,
    col2 INT
  ) ENGINE NDB;

  CREATE TABLE child(
    ref1 INT UNIQUE,
    FOREIGN KEY fk1(ref1)
      REFERENCES parent(col1) ON UPDATE CASCADE
  ) ENGINE NDB;

  CREATE TABLE grandchild(
    ref2 INT,
    FOREIGN KEY fk2(ref2)
      REFERENCES child(ref1) ON UPDATE CASCADE
  ) ENGINE NDB;
  ```

  Table `child` is a child of table `parent`; table `grandchild` is a child of table `child`, and a grandchild of `parent`. In this scenario, a change to column `col1` of `parent` cascaded to `ref1` in table `child`, but it was not always propagated in turn to `ref2` in table `grandchild`. (Bug #83743, Bug #25063506)

- 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)

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

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

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

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

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 5.6 through MySQL 5.6.34 (see Changes in MySQL 5.6.34 (2016-10-12, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

## Functionality Added or Changed

- **MySQL NDB ClusterJ:** To help applications handle database errors better, a number of new features have been added to the `ClusterJDatastoreException` class:
  - A new method, `getCode()`, returns `code` from the `NdbError` object.
  - A new method, `getMysqlCode()`, returns `mysql_code` from the `NdbError` object.
  - A new subclass, `ClusterJDatastoreException.Classification`, gives users the ability to decode the result from `getClassification()`. The method `Classification.toString()` gives the name of the error classification as listed in NDB Error Classifications.

  (Bug #22353594)

## Bugs Fixed

- **NDB Cluster APIs:** Reuse of transaction IDs could occur when `Ndb` objects were created and deleted concurrently. As part of this fix, the NDB API methods `lock_ndb_objects()` and `unlock_ndb_objects` are now declared as `const`. (Bug #23709232)
- **NDB Cluster APIs:** When the management server was restarted while running an MGM API application that continuously monitored events, subsequent events were not reported to the application, with timeouts being returned indefinitely instead of an error.

  This occurred because sockets for event listeners were not closed when restarting `mgmd`. This is fixed by ensuring that event listener sockets are closed when the management server shuts down, causing applications using functions such as `ndb_logevent_get_next()` to receive a read error following the restart. (Bug #19474782)
- Passing a nonexistent node ID to `CREATE NODEGROUP` led to random data node failures. (Bug #23748958)
- `DROP TABLE` followed by a node shutdown and subesequent master takeover—and with the containing local checkpoint not yet complete prior to the takeover—caused the LCP to be ignored, and in some cases, the data node to fail. (Bug #23735996)

  References: See also: Bug #23288252.
- Removed an invalid assertion to the effect that all cascading child scans are closed at the time API connection records are released following an abort of the main transaction. The assertion was invalid because closing of scans in such cases is by design asynchronous with respect to the main transaction, which means that subscans may well take some time to close after the main transaction is closed. (Bug #23709284)
- A number of potential buffer overflow issues were found and fixed in the `NDB` codebase. (Bug #23152979)

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

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

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

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

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

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

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

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

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

- When restoring a backup taken from a database containing tables that had foreign keys, ndb_restore disabled the foreign keys for data, but not for the logs. (Bug #83155, Bug #24736950)

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

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

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

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

process. Now in such cases, the older, restorable LCP is used, which should help decrease long node restart times. (Bug #81894, Bug #23602217)

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

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

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

  References: See also: Bug #23048816.

- Error and abort handling in `Dbspj::execTRANSID_AI()` was implemented such that its `abort()` method was called before processing of the incoming signal was complete. Since this method sends signals to the LDM, this partly overwrote the contents of the signal which was later required by `execTRANSID_AI()`. This could result in aborted `DBSPJ` requests cleaning up their allocated resources too early, or not at all. (Bug #81353, Bug #23251145)

  References: See also: Bug #23048816.

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

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

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

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

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

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

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 5.6 through MySQL 5.6.31 (see Changes in MySQL 5.6.31 (2016-06-02, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

## Functionality Added or Changed

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

## Bugs Fixed

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

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

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

- **MySQL NDB ClusterJ:** Time value of a `java.sql.Timestamp` object became incorrect when Clusterj stored it into a TIMESTAMP column with fractional seconds in a database table. (Bug #23155061)

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

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

  References: See also: Bug #21098142.

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

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

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

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

  References: See also: Bug #23221573.

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

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

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

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

References: See also: Bug #23343739.

- `DBDICT` and `GETTABINFOREQ` queue debugging were enhanced as follows:
  - Monitoring by a data node of the progress of `GETTABINFOREQ` signals can be enabled by setting `DictTrace` >= 2.
  - Added the `ApiVerbose` configuration parameter, which enables NDB API debug logging for an API node where it is set greater than or equal to 2.
  - Added `DUMP` code 1229 which shows the current state of the `GETTABINFOREQ` queue. (See DUMP 1229.)

  See also The DBDICT Block. (Bug #20368450)

  References: See also: Bug #20368354.

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

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

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

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

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 5.6 through MySQL 5.6.29 (see Changes in MySQL 5.6.29 (2016-02-05, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

## Functionality Added or Changed

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

## Bugs Fixed

- **Important Change:** The minimum value for the `BackupDataBufferSize` data node configuration parameter has been lowered from 2 MB to 512 KB. The default and maximum values for this parameter remain unchanged. (Bug #22749509)
- **OS X:** Processing of local checkpoints was not handled correctly on Mac OS X, due to an uninitialized variable. (Bug #80236, Bug #22647462)
- **Microsoft Windows:** Compilation of MySQL with Visual Studio 2015 failed in `ConfigInfo.cpp`, due to a change in Visual Studio's handling of spaces and concatenation. (Bug #22558836, Bug #80024)
- **Microsoft Windows:** When setting up event logging for `ndb_mgmd` on Windows, MySQL NDB Cluster tries to add a registry key to `HKEY_LOCAL_MACHINE`, which fails if the user does not have access to the registry. In such cases `ndb_mgmd` logged the error `Could neither create or open key`, which is not accurate and which can cause confusion for users who may not realize that file logging is available and being used. Now in such cases, `ndb_mgmd` logs a warning `Could`

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

- **Microsoft Windows:** MySQL NDB Cluster did not compile correctly with Microsoft Visual Studio 2015, due to a change from previous versions in the VS implementation of the `_vsnprintf()` function. (Bug #80276, Bug #22670525)
- **Microsoft Windows:** Performing `ANALYZE TABLE` on a table having one or more indexes caused `ndbmtd` to fail with an *`InvalidAttrInfo`* error due to signal corruption. This issue occurred consistently on Windows, but could also be encountered on other platforms. (Bug #77716, Bug #21441297)
- **Solaris:** The `ndb_print_file` utility failed consistently on Solaris 9 for SPARC. (Bug #80096, Bug #22579581)
- **NDB Cluster APIs:** Executing a transaction with an `NdbIndexOperation` based on an obsolete unique index caused the data node process to fail. Now the index is checked in such cases, and if it cannot be used the transaction fails with an appropriate error. (Bug #79494, Bug #22299443)
- Integer overflow could occur during client handshake processing, leading to a server exit. (Bug #22722946)
- During node failure handling, the request structure used to drive the cleanup operation was not maintained correctly when the request was executed. This led to inconsistencies that were harmless during normal operation, but these could lead to assertion failures during node failure handling, with subsequent failure of additional nodes. (Bug #22643129)
- The previous fix for a lack of mutex protection for the internal `TransporterFacade::deliver_signal()` function was found to be incomplete in some cases. (Bug #22615274)

  References: This issue is a regression of: Bug #77225, Bug #21185585.
- When setup of the binary log as an atomic operation on one SQL node failed, this could trigger a state in other SQL nodes in which they appeared to detect the SQL node participating in schema change distribution, whereas it had not yet completed binary log setup. This could in turn cause a deadlock on the global metadata lock when the SQL node still retrying binary log setup needed this lock, while another mysqld had taken the lock for itself as part of a schema change operation. In such cases, the second SQL node waited for the first one to act on its schema distribution changes, which it was not yet able to do. (Bug #22494024)
- For busy servers, client connection or communication failure could occur if an I/O-related system call was interrupted. The `mysql_options()` C API function now has a `MYSQL_OPT_RETRY_COUNT` option to control the number of retries for interrupted system calls. (Bug #22336527)

  References: See also: Bug #22389653.
- Duplicate key errors could occur when `ndb_restore` was run on a backup containing a unique index. This was due to the fact that, during restoration of data, the database can pass through one or more inconsistent states prior to completion, such an inconsistent state possibly having duplicate values for a column which has a unique index. (If the restoration of data is preceded by a run with `--disable-indexes` and followed by one with `--rebuild-indexes`, these errors are avoided.)

  Added a check for unique indexes in the backup which is performed only when restoring data, and which does not process tables that have explicitly been excluded. For each unique index found, a warning is now printed. (Bug #22329365)
- Restoration of metadata with `ndb_restore -m` occasionally failed with the error message `Failed to create index...` when creating a unique index. While disgnosing this problem, it was

found that the internal error **PREPARE_SEIZE_ERROR** (a temporary error) was reported as an unknown error. Now in such cases, `ndb_restore` retries the creation of the unique index, and **PREPARE_SEIZE_ERROR** is reported as NDB Error 748 `Busy during read of event table`. (Bug #21178339)

References: See also: Bug #22989944.

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

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

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

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

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

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

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

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

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

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

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

- When using `CREATE INDEX` to add an index on either of two `NDB` tables sharing circular foreign keys, the query succeeded but a temporary table was left on disk, breaking the foreign key constraints. This issue was also observed when attempting to create an index on a table in the

middle of a chain of foreign keys—that is, a table having both parent and child keys, but on different tables. The problem did not occur when using `ALTER TABLE` to perform the same index creation operation; and subsequent analysis revealed unintended differences in the way such operations were performed by `CREATE INDEX`.

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

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

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

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

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

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

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

## Bugs Fixed

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

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

**Note**

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

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

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

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 5.6 through MySQL 5.6.28 (see Changes in MySQL 5.6.28 (2015-12-07, General Availability)).

## Functionality Added or Changed

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

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

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

## Bugs Fixed

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

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

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

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

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

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

  When the binary log inject thread receives a group of one or more failure events, it drops all other existing event operations and expects no more events from the utility thread until it has handled all of the failure events and then restarted itself. However, it was possible for the utility thread to continue attempting binary log setup while the injector thread was handling failures and thus attempting to

create the schema distribution tables as well as event subscriptions on these tables. If the creation of these tables and event subscriptions occurred during this time, the binary log injector thread's expectation that there were no further event operations was never met; thus, the injector thread never restarted, and NDB tables remained in read-only as described previously.

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

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

  References: See also: Bug #20646496.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 5.6 through MySQL 5.6.27 (see Changes in MySQL 5.6.27 (2015-09-30, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

## Functionality Added or Changed

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

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

    - BackupDataBufferSize: minimum increased from 0 to 2M.
    - BackupLogBufferSize: minimum increased from 0 to 2M.
    - BackupWriteSize: minimum increased from 2K to 32K.
    - BackupMaxWriteSize: minimum increased from 2K to 256K.

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

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

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

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

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

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

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

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

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

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

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

## Bugs Fixed

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- The loopback transporter is similar to the TCP transporter, but is used by a node to send signals to itself as part of many internal operations. Like the TCP transporter, it could be disconnected due to certain conditions including send buffer exhaustion, but this could result in blocking of `TransporterFacade` and so cause multiple issues within an `ndb_mgmd` or API node process. To

prevent this, a node whose loopback transporter becomes disconnected is now simply shut down, rather than allowing the node process to hang. (Bug #21651400, Bug #22016081)

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

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

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

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

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

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

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

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

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

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

  References: See also: Bug #21372136.

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

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

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

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

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

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

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

- The `inet_ntoa()` function used internally in several `mgmd` threads was not `POSIX` thread-safe, which meant that the result it returned could sometimes be undefined. To avoid this problem, a thread-safe and platform-independent wrapper for `inet_ntop()` is used to take the place of this function. (Bug #17766129)

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

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

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

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

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

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

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

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

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

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

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 5.6 through MySQL 5.6.25 (see Changes in MySQL 5.6.25 (2015-05-29, General Availability)).

- Functionality Added or Changed

- Bugs Fixed

## Functionality Added or Changed

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## Bugs Fixed

- **Incompatible Change; NDB Cluster APIs:** The `pollEvents2()` method now returns -1, indicating an error, whenever a negative value is used for the time argument. (Bug #20762291)
- **Important Change; NDB Cluster APIs:** The `Ndb::getHighestQueuedEpoch()` method returned the greatest epoch in the event queue instead of the greatest epoch found after calling `pollEvents2()`. (Bug #20700220)
- **Important Change; NDB Cluster APIs:** `Ndb::pollEvents()` is now compatible with the `TE_EMPTY`, `TE_INCONSISTENT`, and `TE_OUT_OF_MEMORY` event types introduced in MySQL NDB Cluster 7.4.3. For detailed information about this change, see the description of this method in the *MySQL NDB Cluster API Developer Guide*. (Bug #20646496)
- **Important Change; NDB Cluster APIs:** Added the method `Ndb::isExpectingHigherQueuedEpochs()` to the NDB API to detect when additional, newer event epochs were detected by `pollEvents2()`.

  The behavior of `Ndb::pollEvents()` has also been modified such that it now returns **`NDB_FAILURE_GCI`** (equal to `~(Uint64) 0`) when a cluster failure has been detected. (Bug #18753887)
- **NDB Cluster APIs:** Added the `Column::getSizeInBytesForRecord()` method, which returns the size required for a column by an `NdbRecord`, depending on the column's type (text/blob, or other). (Bug #21067283)
- **NDB Cluster APIs:** `NdbEventOperation::isErrorEpoch()` incorrectly returned `false` for the `TE_INCONSISTENT` table event type (see Event::TableEvent). This caused a subsequent call to `getEventType()` to fail. (Bug #20729091)
- **NDB Cluster APIs:** Creation and destruction of `Ndb_cluster_connection` objects by multiple threads could make use of the same application lock, which in some cases led to failures in the global dictionary cache. To alleviate this problem, the creation and destruction of several internal NDB API objects have been serialized. (Bug #20636124)
- **NDB Cluster APIs:** A number of timeouts were not handled correctly in the NDB API. (Bug #20617891)
- **NDB Cluster APIs:** When an `Ndb` object created prior to a failure of the cluster was reused, the event queue of this object could still contain data node events originating from before the failure. These events could reference “old” epochs (from before the failure occurred), which in turn could violate the assumption made by the `nextEvent()` method that epoch numbers always increase. This issue is addressed by explicitly clearing the event queue in such cases. (Bug #18411034)

  References: See also: Bug #20888668.
- **MySQL NDB ClusterJ:** When used with Java 1.7 or higher, ClusterJ might cause the Java VM to crash when querying tables with BLOB columns, because `NdbDictionary::createRecord` calculates the wrong size needed for the record. Subsequently, when ClusterJ called `NdbScanOperation::nextRecordCopyOut`, the data overran the allocated buffer space. With this fix, ClusterJ checks the size calculated by `NdbDictionary::createRecord` and uses the value for the buffer size, if it is larger than the value ClusterJ itself calculates. (Bug #20695155)
- After restoring the database metadata (but not any data) by running `ndb_restore --restore-meta` (or `-m`), SQL nodes would hang while trying to `SELECT` from a table in the database to which the metadata was restored. In such cases the attempt to query the table now fails as expected, since

the table does not actually exist until `ndb_restore` is executed with `--restore-data` (`-r`). (Bug #21184102)

References: See also: Bug #16890703.

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

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

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

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

  References: See also: Bug #20481140.

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

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

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

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

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

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

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

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

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

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

  References: See also: Bug #17916243.

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

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

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

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

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

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

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

  (Bug #77081, Bug #21109605)

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

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

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

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

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 5.6 through MySQL 5.6.24 (see Changes in MySQL 5.6.24 (2015-04-06, General Availability)).

## Bugs Fixed

- During backup, loading data from one SQL node followed by repeated `DELETE` statements on the tables just loaded from a different SQL node could lead to data node failures. (Bug #18949230)
- When an instance of `NdbEventBuffer` was destroyed, any references to GCI operations that remained in the event buffer data list were not freed. Now these are freed, and items from the event bufer data list are returned to the free list when purging GCI containers. (Bug #76165, Bug #20651661)
- When a bulk delete operation was committed early to avoid an additional round trip, while also returning the number of affected rows, but failed with a timeout error, an SQL node performed no verification that the transaction was in the Committed state. (Bug #74494, Bug #20092754)

  References: See also: Bug #19873609.

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

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

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

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

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 5.6 through MySQL 5.6.23 (see Changes in MySQL 5.6.23 (2015-02-02, General Availability)).

## Bugs Fixed

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

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

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

- **NDB Cluster APIs:** A scan operation, whether it is a single table scan or a query scan used by a pushed join, stores the result set in a buffer. This maximum size of this buffer is calculated and preallocated before the scan operation is started. This buffer may consume a considerable amount of memory; in some cases we observed a 2 GB buffer footprint in tests that executed 100 parallel scans

with 2 single-threaded (ndbd) data nodes. This memory consumption was found to scale linearly with additional fragments.

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

- Result rows were unpacked to full NdbRecord format before they were stored in the buffer. If only some but not all columns of a table were selected, the buffer contained empty space (essentially wasted).
- Due to the buffer format being unpacked, VARCHAR and VARBINARY columns always had to be allocated for the maximum size defined for such columns.
- BatchByteSize and MaxScanBatchSize values were not taken into consideration as a limiting factor when calculating the maximum buffer size.

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

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

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

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

  References: See also: Bug #34702.

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

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

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

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

  References: See also: Bug #44529.

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

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

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

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

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

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

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

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

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

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

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 5.6 through MySQL 5.6.23 (see Changes in MySQL 5.6.23 (2015-02-02, General Availability)).

## Bugs Fixed

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 5.6 through MySQL 5.6.22 (see Changes in MySQL 5.6.22 (2014-12-01, General Availability)).

- Functionality Added or Changed

- Bugs Fixed

## Functionality Added or Changed

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

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

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

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

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

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

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

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

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

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

## Bugs Fixed

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

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

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

  References: See also: Bug #19846392.

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

  References: See also: Bug #19999242.

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

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

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

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

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

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

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

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

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

- When a new node failed after connecting to the president but not to any other live node, then reconnected and started again, a live node that did not see the original connection retained old state information. This caused the live node to send redundant signals to the president, causing it to fail. (Bug #75218, Bug #20215395)
- In the `NDB` kernel, it was possible for a `TransporterFacade` object to reset a buffer while the data contained by the buffer was being sent, which could lead to a race condition. (Bug #75041, Bug #20112981)
- `mysql_upgrade` failed to drop and recreate the `ndbinfo` database and its tables as expected. (Bug #74863, Bug #20031425)
- Due to a lack of memory barriers, MySQL NDB Cluster programs such as `ndbmtd` did not compile on `POWER` platforms. (Bug #74782, Bug #20007248)
- In spite of the presence of a number of protection mechanisms against overloading signal buffers, it was still in some cases possible to do so. This fix adds block-level support in the `NDB` kernel (in `SimulatedBlock`) to make signal buffer overload protection more reliable than when implementing such protection on a case-by-case basis. (Bug #74639, Bug #19928269)
- Copying of metadata during local checkpoints caused node restart times to be highly variable which could make it difficult to diagnose problems with restarts. The fix for this issue introduces signals (including `PAUSE_LCP_IDLE`, `PAUSE_LCP_REQUESTED`, and `PAUSE_NOT_IN_LCP_COPY_META_DATA`) to pause LCP execution and flush LCP reports, making it possible to block LCP reporting at times when LCPs during restarts become stalled in this fashion. (Bug #74594, Bug #19898269)
- When a data node was restarted from its angel process (that is, following a node failure), it could be allocated a new node ID before failure handling was actually completed for the failed node. (Bug #74564, Bug #19891507)
- In `NDB` version 7.4, node failure handling can require completing checkpoints on up to 64 fragments. (This checkpointing is performed by the `DBLQH` kernel block.) The requirement for master takeover to wait for completion of all such checkpoints led in such cases to excessive length of time for completion.

  To address these issues, the `DBLQH` kernel block can now report that it is ready for master takeover before it has completed any ongoing fragment checkpoints, and can continue processing these while the system completes the master takeover. (Bug #74320, Bug #19795217)
- Local checkpoints were sometimes started earlier than necessary during node restarts, while the node was still waiting for copying of the data distribution and data dictionary to complete. (Bug #74319, Bug #19795152)
- The check to determine when a node was restarting and so know when to accelerate local checkpoints sometimes reported a false positive. (Bug #74318, Bug #19795108)
- Values in different columns of the `ndbinfo` tables `disk_write_speed_aggregate` and `disk_write_speed_aggregate_node` were reported using differing multiples of bytes. Now all of these columns display values in bytes.

  In addition, this fix corrects an error made when calculating the standard deviations used in the `std_dev_backup_lcp_speed_last_10sec`, `std_dev_redo_speed_last_10sec`, `std_dev_backup_lcp_speed_last_60sec`, and `std_dev_redo_speed_last_60sec` columns of the `ndbinfo.disk_write_speed_aggregate` table. (Bug #74317, Bug #19795072)
- Recursion in the internal method `Dblqh::finishScanrec()` led to an attempt to create two list iterators with the same head. This regression was introduced during work done to optimize scans for version 7.4 of the `NDB` storage engine. (Bug #73667, Bug #19480197)

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

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

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

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

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

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 5.6 through MySQL 5.6.21 (see Changes in MySQL 5.6.21 (2014-09-23, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

## Functionality Added or Changed

- Added the `restart_info` table to the `ndbinfo` information database to provide current status and timing information relating to node and system restarts. By querying this table, you can observe the progress of restarts in real time. (Bug #19795152)
- After adding new data nodes to the configuration file of a MySQL NDB Cluster having many API nodes, but prior to starting any of the data node processes, API nodes tried to connect to these “missing” data nodes several times per second, placing extra loads on management nodes and the network. To reduce unnecessary traffic caused in this way, it is now possible to control the amount of time that an API node waits between attempts to connect to data nodes which fail to respond; this is implemented in two new API node configuration parameters `StartConnectBackoffMaxTime` and `ConnectBackoffMaxTime`.

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

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

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

## Bugs Fixed

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

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

- When performing a batched update, where one or more successful write operations from the start of the batch were followed by write operations which failed without being aborted (due to the

`AbortOption` being set to `AO_IgnoreError`), the failure handling for these by the transaction coordinator leaked `CommitAckMarker` resources. (Bug #19875710)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 5.6 through MySQL 5.6.20 (see Changes in MySQL 5.6.20 (2014-07-31, General Availability)).



## Conflict Resolution Exceptions Table Extensions

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

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

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

  Read tracking—that is, detecting conflicts between reads of a given row in one cluster and updates or deletes of the same row in another cluster—is now supported. This requires exclusive read locks obtained by setting `ndb_log_exclusive_reads` equal to 1 on the slave cluster. All rows read by a conflicting read are logged in the exceptions table. For more information and examples, see Read conflict detection and resolution.

  Existing exceptions tables continue to be supported. For additional information, see Conflict Resolution Exceptions Table.

## Node Restart Performance and Reporting Enhancements

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

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

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

  - Previously, there could be at most 2 LDM instances active with local checkpoints at any given time. Now, up to 16 LDMs can be used for performing this task, which increases utilization of

available CPU power, and can speed up LCPs by a factor of 10, which in turn can greatly improve restart times.

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

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

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

## Dynamic Primary/Secondary Role Determination

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

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

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

## Improved Scan and SQL Processing

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

## Per-Fragment Memory Reporting

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

## Bugs Fixed

- **NDB Cluster APIs:** When an NDB API client application received a signal with an invalid block or signal number, `NDB` provided only a very brief error message that did not accurately convey the

nature of the problem. Now in such cases, appropriate printouts are provided when a bad signal or message is detected. In addition, the message length is now checked to make certain that it matches the size of the embedded signal. (Bug #18426180)

- In some cases, transporter receive buffers were reset by one thread while being read by another. This happened when a race condition occurred between a thread receiving data and another thread initiating disconnect of the transporter (disconnection clears this buffer). Concurrency logic has now been implemented to keep this race from taking place. (Bug #19552283, Bug #73790)
- When a new data node started, API nodes were allowed to attempt to register themselves with the data node for executing transactions before the data node was ready. This forced the API node to wait an extra heartbeat interval before trying again.

  To address this issue, a number of **HA_ERR_NO_CONNECTION** errors (Error 4009) that could be issued during this time have been changed to `Cluster temporarily unavailable` errors (Error 4035), which should allow API nodes to use new data nodes more quickly than before. As part of this fix, some errors which were incorrectly categorised have been moved into the correct categories, and some errors which are no longer used have been removed. (Bug #19524096, Bug #73758)
- Executing `ALTER TABLE ... REORGANIZE PARTITION` after increasing the number of data nodes in the cluster from 4 to 16 led to a crash of the data nodes. This issue was shown to be a regression caused by previous fix which added a new dump handler using a dump code that was already in use (7019), which caused the command to execute two different handlers with different semantics. The new handler was assigned a new `DUMP` code (7024). (Bug #18550318)

  References: This issue is a regression of: Bug #14220269.
- When certain queries generated signals having more than 18 data words prior to a node failure, such signals were not written correctly in the trace file. (Bug #18419554)
- Failure of multiple nodes while using `ndbmtd` with multiple TC threads was not handled gracefully under a moderate amount of traffic, which could in some cases lead to an unplanned shutdown of the cluster. (Bug #18069334)
- For multithreaded data nodes, some threads do communicate often, with the result that very old signals can remain at the top of the signal buffers. When performing a thread trace, the signal dumper calculated the latest signal ID from what it found in the signal buffers, which meant that these old signals could be erroneously counted as the newest ones. Now the signal ID counter is kept as part of the thread state, and it is this value that is used when dumping signals for trace files. (Bug #73842, Bug #19582807)

# Release Series Changelogs: MySQL NDB Cluster 7.4

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

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

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

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

## Changes in MySQL NDB Cluster 7.4.35 (5.6.51-ndb-7.4.35) (2022-01-19, General Availability)

### Bugs Fixed

- **NDB Cluster APIs:** It is no longer possible to use the `DIVERIFYREQ` signal asynchronously. (Bug #33161562)

- Added missing values checks in `ndbd` and `ndbmtd`. (Bug #33661024)
- In certain cases, an event's category was not properly detected. (Bug #33304814)
- `DBDICT` did not always perform table name checks correctly. (Bug #33161548)
- `SET_LOGLEVELORD` signals were not always handled correctly. (Bug #33161246)
- `DUMP 11001` did not always handle all of its arguments correctly. (Bug #33157513)
- File names were not always verified correctly. (Bug #33157475)
- Added a number of missing ID and other values checks in `ndbd` and `ndbmtd`. (Bug #32983700, Bug #32893708, Bug #32957478, Bug #32983256, Bug #32983339, Bug #32983489, Bug #32983517, Bug #33157527, Bug #33157531, Bug #33161271, Bug #33161298, Bug #33161314, Bug #33161331, Bug #33161372, Bug #33161462, Bug #33161511, Bug #33161519, Bug #33161537, Bug #33161570, Bug #33162059, Bug #33162065, Bug #33162074, Bug #33162082, Bug #33162092, Bug #33162098, Bug #33304819)
- The management server did not always handle events of the wrong size correctly. (Bug #32957547)

## Changes in MySQL NDB Cluster 7.4.34 (5.6.51-ndb-7.4.34) (2021-10-20, General Availability)

### Bugs Fixed

- NDB Cluster could not be compiled using GCC 10 or 11. (Bug #33282549)
- A buffer used in the `SUMA` kernel block did not always accommodate multiple signals. (Bug #33246047)
- Added an `ndbrequire()` in `QMGR` to check whether the node ID received from the `CM_REGREF` signal is less than `MAX_NDB_NODES`. (Bug #32983311)
- A check was reported missing from the code for handling `GET_TABLEID_REQ` signals. To fix this issue, all code relating to all `GET_TABLEID_*` signals has been removed from the `NDB` sources, since these signals are no longer used or supported in NDB Cluster. (Bug #32983249)
- It was possible in some cases to specify an invalid node type when working with the internal management API. Now the API specifically disallows invalid node types, and defines an “unknown” node type (`NDB_MGM_NODE_TYPE_UNKNOWN`) to cover such cases. (Bug #32957364)
- `ndb_restore` raised a warning to use `--disable-indexes` when restoring data after the metadata had already been restored with `--disable-indexes`.

  When `--disable-indexes` is used to restore metadata before restoring data, the tables in the target schema have no indexes. We now check when restoring data with this option to ensure that there are no indexes on the target table, and print the warning only if the table already has indexes. (Bug #28749799)
- When restoring of metadata was done using `--disable-indexes`, there was no attempt to create indexes or foreign keys dependent on these indexes, but when `ndb_restore` was used without the option, indexes and foreign keys were created. When `--disable-indexes` was used later while restoring data, `NDB` attempted to drop any indexes created in the previous step, but ignored the failure of a drop index operation due to a dependency on the index of a foreign key which had not been dropped. This led subsequently to problems while rebuilding indexes, when there was an attempt to create foreign keys which already existed.

  We fix `ndb_restore` as follows:

  - When `--disable-indexes` is used, `ndb_restore` now drops any foreign keys restored from the backup.

- `ndb_restore` now checks for the existence of indexes before attempting to drop them.

(Bug #26974491)

## Changes in MySQL NDB Cluster 7.4.33 (5.6.51-ndb-7.4.33) (2021-07-21, General Availability)

### Bugs Fixed

- **Packaging:** The `ndb-common` man page was removed, and the information it contained moved to other man pages. (Bug #32799519)
- `Ndb_rep_tab_key` member variables were not null-terminated before being logged. (Bug #32841430)

  References: See also: Bug #32393245.

## Changes in MySQL NDB Cluster 7.4.31 (5.6.51-ndb-7.4.31) (2021-01-19, General Availability)

- Deprecation and Removal Notes
- Bugs Fixed

### Deprecation and Removal Notes

- **NDB Client Programs:** Effective with this release, the MySQL NDB Cluster Auto-Installer (`ndb_setup.py`) has been has been removed from the NDB Cluster binary and source distributions, and is no longer supported. (Bug #32084831)

  References: See also: Bug #31888835.

### Bugs Fixed

- Using the maximum size of an index key supported by index statistics (3056 bytes) caused buffer issues in data nodes. (Bug #32094904)

  References: See also: Bug #25038373.

- When a table creation schema transaction is prepared, the table is in `TS_CREATING` state, and is changed to `TS_ACTIVE` state when the schema transaction commits on the `DBDIH` block. In the case where the node acting as `DBDIH` coordinator fails while the schema transaction is committing, another node starts taking over for the coordinator. The following actions are taken when handling this node failure:

  - `DBDICT` rolls the table creation schema transaction forward and commits, resulting in the table involved changing to `TS_ACTIVE` state.
  - `DBDIH` starts removing the failed node from tables by moving active table replicas on the failed node from a list of stored fragment replicas to another list.

  These actions are performed asynchronously many times, and when interleaving may cause a race condition. As a result, the replica list in which the replica of a failed node resides becomes nondeterministic and may differ between the recovering node (that is, the new coordinator) and other `DIH` participant nodes. This difference violated a requirement for knowing which list the failed node's replicas can be found during the recovery of the failed node recovery on the other participants.

  To fix this, moving active table replicas now covers not only tables in `TS_ACTIVE` state, but those in `TS_CREATING` (prepared) state as well, since the prepared schema transaction is always rolled forward.

In addition, the state of a table creation schema transaction which is being aborted is now changed from `TS_CREATING` or `TS_IDLE` to `TS_DROPPING`, to avoid any race condition there. (Bug #30521812)

# Changes in MySQL NDB Cluster 7.4.30 (5.6.50-ndb-7.4.30) (2020-10-20, General Availability)

- Deprecation and Removal Notes
- Bugs Fixed

## Deprecation and Removal Notes

- **NDB Cluster APIs:** Support for Node.js has been removed in this release.

  Node.js continues to be supported in NDB Cluster 8.0 only. (Bug #31781948)

- **NDB Client Programs:** Effective with this release, the MySQL NDB Cluster Auto-Installer (`ndb_setup.py`) has been deprecated and is subject to removal in a future version of NDB Cluster. (Bug #31888835)

## Bugs Fixed

- **Packaging:** The Dojo library included with NDB Cluster has been upgraded to version 1.15.4. (Bug #31559518)

# Changes in MySQL NDB Cluster 7.4.29 (5.6.49-ndb-7.4.29) (2020-07-14, General Availability)

## Bugs Fixed

- During a node restart, the `SUMA` block of the node that is starting must get a copy of the subscriptions (events with subscribers) and subscribers (`NdbEventOperation` instances which are executing) from a node already running. Before the copy is complete, nodes which are still starting ignore any user-level `SUB_START` or `SUB_STOP` requests; after the copy is done, they can participate in such requests. While the copy operation is in progress, user-level `SUB_START` and `SUB_STOP` requests are blocked using a `DICT` lock.

  An issue was found whereby a starting node could participate in `SUB_START` and `SUB_STOP` requests after the lock was requested, but before it is granted, which resulted in unsuccessful `SUB_START` and `SUB_STOP` requests. This fix ensures that the nodes cannot participate in these requests until after the `DICT` lock has actually been granted. (Bug #31302657)

- The Dojo toolkit included with NDB Cluster and used by the Auto-Installer was upgraded to version 1.15.3. (Bug #31029110)

- A packed version 1 configuration file returned by `ndb_mgmd` could contain duplicate entries following an upgrade to NDB 8.0, which made the file incompatible with clients using version 1. This occurs due to the fact that the code for handling backwards compatibility assumed that the entries in each section were already sorted when merging it with the default section. To fix this, we now make sure that this sort is performed prior to merging. (Bug #31020183)

- When executing any of the `SHUTDOWN`, `ALL STOP`, or `ALL RESTART` management commands, it is possible for different nodes to attempt to stop on different global checkpoint index (CGI) boundaries. If they succeed in doing so, then a subsequent system restart is slower than normal because any nodes having an earlier stop GCI must undergo takeover as part of the process. When nodes failing on the first GCI boundary cause surviving nodes to be nonviable, surviving nodes suffer an arbitration failure; this has the positive effect of causing such nodes to halt at the correct GCI, but can give rise to spurious errors or similar.

To avoid such issues, extra synchronization is now performed during a planned shutdown to reduce the likelihood that different data nodes attempt to shut down at different GCIs as well as the use of unnecessary node takeovers during system restarts. (Bug #31008713)

## Changes in MySQL NDB Cluster 7.4.28 (5.6.48-ndb-7.4.28) (2020-04-28, General Availability)

- Functionality Added or Changed
- Bugs Fixed

### Functionality Added or Changed

- Added the `--ndb-log-fail-terminate` option for `mysqld`. When used, this causes the SQL node to terminate if it is unable to log all row events. (Bug #21911930)

  References: See also: Bug #30383919.

### Bugs Fixed

- When a node ID allocation request failed with **`NotMaster`** temporary errors, the node ID allocation was always retried immediately, without regard to the cause of the error. This caused a very high rate of retries, whose effects could be observed as an excessive number of `Alloc node id for node nnn failed` log messages (on the order of 15,000 messages per second). (Bug #30293495)
- For `NDB` tables having no explicit primary key, `NdbReceiverBuffer` could be allocated with too small a size. This was due to the fact that the attribute bitmap sent to `NDB` from the data nodes always includes the primary key. The extra space required for hidden primary keys is now taken into consideration in such cases. (Bug #30183466)

## Changes in MySQL NDB Cluster 7.4.27 (5.6.47-ndb-7.4.27) (2020-01-14, General Availability)

### Bugs Fixed

- If a transaction was aborted while getting a page from the disk page buffer and the disk system was overloaded, the transaction hung indefinitely. This could also cause restarts to hang and node failure handling to fail. (Bug #30397083, Bug #30360681)

  References: See also: Bug #30152258.

- The maximum global checkpoint (GCP) commit lag and GCP save timeout are recalculated whenever a node shuts down, to take into account the change in number of data nodes. This could lead to the unintentional shutdown of a viable node when the threshold decreased below the previous value. (Bug #27664092)

  References: See also: Bug #26364729.

- Concurrent `SELECT` and `ALTER TABLE` statements on the same SQL node could sometimes block one another while waiting for locks to be released. (Bug #17812505, Bug #30383887)

## Changes in MySQL NDB Cluster 7.4.26 (5.6.46-ndb-7.4.26) (2019-10-15, General Availability)

### Bugs Fixed

- During a restart when the data nodes had started but not yet elected a president, the management server received a `node ID already in use` error, which resulted in excessive retries and

logging. This is fixed by introducing a new error 1705 `Not ready for connection allocation yet` for this case.

During a restart when the data nodes had not yet completed node failure handling, a spurious `Failed to allocate nodeID` error was returned. This is fixed by adding a check to detect an incomplete node start and to return error 1703 `Node failure handling not completed` instead.

As part of this fix, the frequency of retries has been reduced for `not ready to alloc nodeID` errors, an error insert has been added to simulate a slow restart for testing purposes, and log messages have been reworded to indicate that the relevant node ID allocation errors are minor and only temporary. (Bug #27484514)

# Changes in MySQL NDB Cluster 7.4.25 (5.6.45-ndb-7.4.25) (2019-07-23, General Availability)

## Bugs Fixed

- The `requestInfo` fields for the long and short forms of the `LQHKEYREQ` signal had different definitions; bits used for the key length in the short version were reused for flags in the long version, since the key length is implicit in the section length of the long version of the signal but it was possible for long `LQHKEYREQ` signals to contain a keylength in these same bits, which could be misinterpreted by the receiving local query handler, potentially leading to errors. Checks have now been implemented to make sure that this no longer happens. (Bug #29820838)
- When restoring `TINYBLOB` columns, `ndb_restore` now treats them as having the `BINARY` character set. (Bug #29486538)
- Restoration of epochs by `ndb_restore` failed due to temporary redo errors. Now `ndb_restore` retries epoch updates when such errors occur. (Bug #29466089)
- `ndb_restore --restore-epoch` incorrectly reported the stop GCP as 1 less than the actual position. (Bug #29343655)
- Added support which was missing in `ndb_restore` for conversions between the following sets of types:
  - `BLOB` and `BINARY` or `VARBINARY` columns
  - `TEXT` and `BLOB` columns
  - `BLOB` columns with unequal lengths
  - `BINARY` and `VARBINARY` columns with unequal lengths

  (Bug #28074988)

# Changes in MySQL NDB Cluster 7.4.24 (5.6.44-ndb-7.4.24) (2019-04-26, General Availability)



## Functionality Added or Changed

- Building with `CMake3` is now supported by the `compile-cluster` script included in the `NDB` source distribution.

### Bugs Fixed

- **Important Change:** The dependency of `ndb_restore` on the `NDBT` library, which is used for internal testing only, has been removed. This means that the program no longer prints `NDBT_ProgramExit: ...` when terminating. Applications that depend upon this behavior should be updated to reflect this change when upgrading to this release.

- When a pushed join executing in the `DBSPJ` block had to store correlation IDs during query execution, memory for these was allocated for the lifetime of the entire query execution, even though these specific correlation IDs are required only when producing the most recent batch in the result set. Subsequent batches require additional correlation IDs to be stored and allocated; thus, if the query took sufficiently long to complete, this led to exhaustion of query memory (error 20008). Now in such cases, memory is allocated only for the lifetime of the current result batch, and is freed and made available for re-use following completion of the batch. (Bug #29336777)

  References: See also: Bug #26995027.

- In some cases, one and sometimes more data nodes underwent an unplanned shutdown while running `ndb_restore`. This occurred most often, but was not always restircted to, when restoring to a cluster having a different number of data nodes from the cluster on which the original backup had been taken.

  The root cause of this issue was exhaustion of the pool of `SafeCounter` objects, used by the `DBDICT` kernel block as part of executing schema transactions, and taken from a per-block-instance pool shared with protocols used for `NDB` event setup and subscription processing. The concurrency of event setup and subscription processing is such that the `SafeCounter` pool can be exhausted; event and subscription processing can handle pool exhaustion, but schema transaction processing could not, which could result in the node shutdown experienced during restoration.

  This problem is solved by giving `DBDICT` schema transactions an isolated pool of reserved `SafeCounters` which cannot be exhausted by concurrent `NDB` event activity. (Bug #28595915)

- `ndb_restore` did not restore autoincrement values correctly when one or more staging tables were in use. As part of this fix, we also in such cases block applying of the `SYSTAB_0` backup log, whose content continued to be applied directly based on the table ID, which could ovewrite the autoincrement values stored in `SYSTAB_0` for unrelated tables. (Bug #27917769, Bug #27831990)

  References: See also: Bug #27832033.

- `ndb_restore` employed a mechanism for restoring autoincrement values which was not atomic, and thus could yield incorrect autoincrement values being restored when multiple instances of `ndb_restore` were used in parallel. (Bug #27832033)

  References: See also: Bug #27917769, Bug #27831990.

- When executing the redo log in debug mode it was possible for a data node to fail when deallocating a row. (Bug #93273, Bug #28955797)

- An `NDB` table having both a foreign key on another `NDB` table using `ON DELETE CASCADE` and one or more `TEXT` or `BLOB` columns leaked memory.

  As part of this fix, `ON DELETE CASCADE` is no longer supported for foreign keys on `NDB` tables when the child table contains a column that uses any of the `BLOB` or `TEXT` types. (Bug #89511, Bug #27484882)

## Changes in MySQL NDB Cluster 7.4.23 (5.6.43-ndb-7.4.23) (2019-01-22, General Availability)

### Bugs Fixed

- **NDB Disk Data:** When a log file group had more than 18 undo logs, it was not possible to restart the cluster. (Bug #251155785)

  References: See also: Bug #28922609.

- When a local checkpoint (LCP) was complete on all data nodes except one, and this node failed, `NDB` did not continue with the steps required to finish the LCP. This led to the following issues:

  No new LCPs could be started.

  Redo and Undo logs were not trimmed and so grew excessively large, causing an increase in times for recovery from disk. This led to write service failure, which eventually led to cluster shutdown when the head of the redo log met the tail. This placed a limit on cluster uptime.

  Node restarts were no longer possible, due to the fact that a data node restart requires that the node's state be made durable on disk before it can provide redundancy when joining the cluster. For a cluster with two data nodes and two fragment replicas, this meant that a restart of the entire cluster (system restart) was required to fix the issue (this was not necessary for a cluster with two fragment replicas and four or more data nodes). (Bug #28728485, Bug #28698831)

  References: See also: Bug #11757421.

- It was possible in certain cases for nodes to hang during an initial restart. (Bug #28698831)

  References: See also: Bug #27622643.

- When tables with `BLOB` columns were dropped and then re-created with a different number of `BLOB` columns the event definitions for monitoring table changes could become inconsistent in certain error situations involving communication errors when the expected cleanup of the corresponding events was not performed. In particular, when the new versions of the tables had more `BLOB` columns than the original tables, some events could be missing. (Bug #27072756)

- When running a cluster with 4 or more data nodes under very high loads, data nodes could sometimes fail with Error 899 `Rowid already allocated`. (Bug #25960230)

- When starting, a data node copies metadata, while a local checkpoint updates metadata. To avoid any conflict, any ongoing LCP activity is paused while metadata is being copied. An issue arose when a local checkpoint was paused on a given node, and another node that was also restarting checked for a complete LCP on this node; the check actually caused the LCP to be completed before copying of metadata was complete and so ended the pause prematurely. Now in such cases, the LCP completion check waits to complete a paused LCP until copying of metadata is finished and the pause ends as expected, within the LCP in which it began. (Bug #24827685)

- Asynchronous disconnection of `mysqld` from the cluster caused any subsequent attempt to start an NDB API transaction to fail. If this occurred during a bulk delete operation, the SQL layer called `HA::end_bulk_delete()`, whose implementation by `ha_ndbcluster` assumed that a transaction had been started, and could fail if this was not the case. This problem is fixed by checking that the transaction pointer used by this method is set before referencing it. (Bug #20116393)

## Changes in MySQL NDB Cluster 7.4.22 (5.6.42-ndb-7.4.22) (2018-10-23, General Availability)

### Bugs Fixed

- 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.

## Changes in MySQL NDB Cluster 7.4.21 (5.6.41-ndb-7.4.21) (2018-07-27, General Availability)

### Bugs Fixed

- **NDB Cluster APIs:** When `Ndb::dropEventOperation()` tried to clean up a pending event, it failed to clear a pointer to the list of GCI operations being deleted and discarded (`Gci_ops` object), so that this pointer referred to a deleted object. GCI operations arriving after this could then be inserted as part of the next such list belonging to the now-deleted object, leading to memory corruption and other issues. (Bug #90011, Bug #27675005)

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

  References: See also: Bug #28698831.

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

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

## Changes in MySQL NDB Cluster 7.4.20 (5.6.40-ndb-7.4.20) (2018-04-20, General Availability)

### Bugs Fixed

- **NDB Cluster APIs:** The maximum time to wait which can be specified when calling either of the NDB API methods `Ndb::pollEvents()` or `pollEvents2()` was miscalculated such that the method could wait up to 9 ms too long before returning to the client. (Bug #88924, Bug #27266086)
- Under certain conditions, data nodes restarted unnecessarily during execution of `ALTER TABLE... REORGANIZE PARTITION`. (Bug #25675481)

  References: See also: Bug #26735618, Bug #27191468.
- Race conditions sometimes occurred during asynchronous disconnection and reconnection of the transporter while other threads concurrently inserted signal data into the send buffers, leading to an unplanned shutdown of the cluster.

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

  References: See also: Bug #20112700.

## Changes in MySQL NDB Cluster 7.4.19 (5.6.39-ndb-7.4.19) (2018-01-23, General Availability)

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

  References: See also: Bug #28728603.
- `ndb_restore` sometimes logged data file and log file progress values much greater than 100%. (Bug #20989106)
- When sending priority A signals, we now ensure that the number of pending signals is explicitly initialized. (Bug #88986, Bug #27294856)
- `ndb_restore --print-data --hex` did not print trailing 0s of `LONGVARBINARY` values. (Bug #65560, Bug #14198580)

## Changes in MySQL NDB Cluster 7.4.18 (5.6.39-ndb-7.4.18) (2018-01-17, General Availability)

### Bugs Fixed

- A query against the `INFORMATION_SCHEMA.FILES` table returned no results when it included an `ORDER BY` clause. (Bug #26877788)
- During a restart, `DBLQH` loads redo log part metadata for each redo log part it manages, from one or more redo log files. Since each file has a limited capacity for metadata, the number of files which must be consulted depends on the size of the redo log part. These files are opened, read, and closed sequentially, but the closing of one file occurs concurrently with the opening of the next.

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

  This issue is resolved by avoiding the use of `OM_WRITE_BUFFER` during metadata reload, so that any transient opening of more than 4 redo log files per log file part no longer leads to failure of the data node. (Bug #25965370)
- Following `TRUNCATE TABLE` on an `NDB` table, its `AUTO_INCREMENT` ID was not reset on an SQL node not performing binary logging. (Bug #14845851)
- When the duplicate weedout algorithm was used for evaluating a semijoin, the result had missing rows. (Bug #88117, Bug #26984919)

  References: See also: Bug #87992, Bug #26926666.
- When representing a materialized semijoin 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 semijoin 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)
- 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 7.4.17 (5.6.38-ndb-7.4.17) (2017-10-18, General Availability)

### Bugs Fixed

- 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.
- 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)

- 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.

## Changes in MySQL NDB Cluster 7.4.16 (5.6.37-ndb-7.4.16) (2017-07-18, General Availability)

- Functionality Added or Changed
- Bugs Fixed

### Functionality Added or Changed

- 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 `--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)

### Bugs Fixed

- **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)

- 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)

- 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)

- 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.

- 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)

- 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)

- 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` or the `ONLINE` keyword. (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)

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

- 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)

- 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)

- 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)

- 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)

- 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.

- 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)

- 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 7.4.15 (5.6.36-ndb-7.4.15) (2017-04-10, General Availability)

### 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.

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

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

- 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)

- 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.

- 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.

- 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 7.4.14 (5.6.35-ndb-7.4.14) (2017-01-17, General Availability)

### Bugs Fixed

- `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.

- Queries against the `ndbinfo.memory_per_fragment` table when running with a large number of data nodes could produce unexpected results for the highest-numbered nodes. (Bug #25176404)

- The `rand()` function was used to produce a unique table ID and table version needed to identify a schema operation distributed between multiple SQL nodes, relying on the assumption that `rand()` would never produce the same numbers on two different instances of mysqld. It was later determined that this is not the case, and that in fact it is very likely for the same random numbers to be produced on all SQL nodes.

  This fix removes the usage of `rand()` for producing a unique table ID or version, and instead uses a sequence in combination with the node ID of the coordinator. This guarantees uniqueness until the counter for the sequence wraps, which should be sufficient for this purpose.

  The effects of this duplication could be observed as timeouts in the log (for example `NDB create db: waiting max 119 sec for distributing`) when restarting multiple `mysqld` processes simultaneously or nearly so, or when issuing the same `CREATE DATABASE` or `DROP DATABASE` statement on multiple SQL nodes. (Bug #24926009)

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

- Long message buffer exhaustion when firing immediate triggers could result in row ID leaks; this could later result in persistent `RowId already allocated` errors (`NDB` Error 899). (Bug #23723110)

References: See also: Bug #19506859, Bug #13927679.

- when a parent `NDB` table in a foreign key relationship was updated, the update cascaded to a child table as expected, but the change was not cascaded to a child table of this child table (that is, to a grandchild of the original parent). This can be illustrated using the tables generated by the following `CREATE TABLE` statements:

```
CREATE TABLE parent(
  id INT PRIMARY KEY AUTO_INCREMENT,
  col1 INT UNIQUE,
  col2 INT
) ENGINE NDB;

CREATE TABLE child(
  ref1 INT UNIQUE,
  FOREIGN KEY fk1(ref1)
    REFERENCES parent(col1) ON UPDATE CASCADE
) ENGINE NDB;

CREATE TABLE grandchild(
  ref2 INT,
  FOREIGN KEY fk2(ref2)
    REFERENCES child(ref1) ON UPDATE CASCADE
) ENGINE NDB;
```

  Table `child` is a child of table `parent`; table `grandchild` is a child of table `child`, and a grandchild of `parent`. In this scenario, a change to column `col1` of `parent` cascaded to `ref1` in table `child`, but it was not always propagated in turn to `ref2` in table `grandchild`. (Bug #83743, Bug #25063506)

- 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)

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

### Bugs Fixed

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

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

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

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

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

  References: See also: Bug #23288252.

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

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

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

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

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

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

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

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

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

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

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

- When restoring a backup taken from a database containing tables that had foreign keys, `ndb_restore` disabled the foreign keys for data, but not for the logs. (Bug #83155, Bug #24736950)

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

- The internal function `ndbcluster_binlog_wait()`, which provides a way to make sure that all events originating from a given thread arrive in the binary log, is used by `SHOW BINLOG EVENTS`

as well as when resetting the binary log. This function waits on an injector condition while the latest global epoch handled by NDB is more recent than the epoch last committed in this session, which implies that this condition must be signalled whenever the binary log thread completes and updates a new latest global epoch. Inspection of the code revealed that this condition signalling was missing, and that, instead of being awakened whenever a new latest global epoch completes (~100ms), client threads waited for the maximum timeout (1 second).

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

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

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

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

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

  References: See also: Bug #23048816.

- Error and abort handling in Dbspj::execTRANSID_AI() was implemented such that its abort() method was called before processing of the incoming signal was complete. Since this method sends signals to the LDM, this partly overwrote the contents of the signal which was later required by execTRANSID_AI(). This could result in aborted DBSPJ requests cleaning up their allocated resources too early, or not at all. (Bug #81353, Bug #23251145)

  References: See also: Bug #23048816.

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

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

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

### Bugs Fixed

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

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

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

- **NDB Cluster APIs:** Deletion of Ndb objects used a dispoportionately high amount of CPU. (Bug #22986823)
- Although arguments to the DUMP command are 32-bit integers, ndb_mgmd used a buffer of only 10 bytes when processing them. (Bug #23708039)
- During shutdown, the mysqld process could sometimes hang after logging NDB Util: Stop ... NDB Util: Wakeup. (Bug #23343739)

  References: See also: Bug #21098142.
- During an online upgrade from a MySQL NDB Cluster 7.3 release to an NDB 7.4 (or later) release, the failures of several data nodes running the lower version during local checkpoints (LCPs), and just prior to upgrading these nodes, led to additional node failures following the upgrade. This was due to lingering elements of the EMPTY_LCP protocol initiated by the older nodes as part of an LCP-plus-restart sequence, and which is no longer used in NDB 7.4 and later due to LCP optimizations implemented in those versions. (Bug #23129433)
- Reserved send buffer for the loopback transporter, introduced in MySQL NDB Cluster 7.4.8 and used by API and management nodes for administrative signals, was calculated incorrectly. (Bug #23093656, Bug #22016081)

  References: This issue is a regression of: Bug #21664515.
- During a node restart, re-creation of internal triggers used for verifying the referential integrity of foreign keys was not reliable, because it was possible that not all distributed TC and LDM instances agreed on all trigger identities. To fix this problem, an extra step is added to the node restart sequence, during which the trigger identities are determined by querying the current master node. (Bug #23068914)

  References: See also: Bug #23221573.
- Following the forced shutdown of one of the 2 data nodes in a cluster where NoOfReplicas=2, the other data node shut down as well, due to arbitration failure. (Bug #23006431)
- The ndbinfo.tc_time_track_stats table uses histogram buckets to give a sense of the distribution of latencies. The sizes of these buckets were also reported as HISTOGRAM BOUNDARY INFO messages during data node startup; this printout was redundant and so has been removed. (Bug #22819868)
- A failure occurred in DBTUP in debug builds when variable-sized pages for a fragment totalled more than 4 GB. (Bug #21313546)
- mysqld did not shut down cleanly when executing ndb_index_stat. (Bug #21098142)

  References: See also: Bug #23343739.
- DBDICT and GETTABINFOREQ queue debugging were enhanced as follows:
  - Monitoring by a data node of the progress of GETTABINFOREQ signals can be enabled by setting DictTrace >= 2.
  - Added the ApiVerbose configuration parameter, which enables NDB API debug logging for an API node where it is set greater than or equal to 2.
  - Added DUMP code 1229 which shows the current state of the GETTABINFOREQ queue. (See DUMP 1229.)

See also The DBDICT Block. (Bug #20368450)

References: See also: Bug #20368354.

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

- Functionality Added or Changed
- Bugs Fixed

## Functionality Added or Changed

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

## Bugs Fixed

- **Important Change:** The minimum value for the `BackupDataBufferSize` data node configuration parameter has been lowered from 2 MB to 512 KB. The default and maximum values for this parameter remain unchanged. (Bug #22749509)
- **OS X:** Processing of local checkpoints was not handled correctly on Mac OS X, due to an uninitialized variable. (Bug #80236, Bug #22647462)
- **Microsoft Windows:** Compilation of MySQL with Visual Studio 2015 failed in `ConfigInfo.cpp`, due to a change in Visual Studio's handling of spaces and concatenation. (Bug #22558836, Bug #80024)
- **Microsoft Windows:** When setting up event logging for `ndb_mgmd` on Windows, MySQL NDB Cluster tries to add a registry key to `HKEY_LOCAL_MACHINE`, which fails if the user does not have access to the registry. In such cases `ndb_mgmd` logged the error `Could neither create or open key`, which is not accurate and which can cause confusion for users who may not realize that file logging is available and being used. Now in such cases, `ndb_mgmd` logs a warning `Could not create or access the registry key needed for the application to log to the Windows EventLog. Run the application with sufficient privileges once to create the key, or add the key manually, or turn off logging for that application.` An error (as opposed to a warning) is now reported in such cases only if there is no available output at all for `ndb_mgmd` event logging. (Bug #20960839)
- **Microsoft Windows:** MySQL NDB Cluster did not compile correctly with Microsoft Visual Studio 2015, due to a change from previous versions in the VS implementation of the `_vsnprintf()` function. (Bug #80276, Bug #22670525)
- **Microsoft Windows:** Performing `ANALYZE TABLE` on a table having one or more indexes caused `ndbmtd` to fail with an *`InvalidAttrInfo`* error due to signal corruption. This issue occurred consistently on Windows, but could also be encountered on other platforms. (Bug #77716, Bug #21441297)
- **Solaris:** The `ndb_print_file` utility failed consistently on Solaris 9 for SPARC. (Bug #80096, Bug #22579581)
- **NDB Cluster APIs:** Executing a transaction with an `NdbIndexOperation` based on an obsolete unique index caused the data node process to fail. Now the index is checked in such cases, and if it cannot be used the transaction fails with an appropriate error. (Bug #79494, Bug #22299443)
- During node failure handling, the request structure used to drive the cleanup operation was not maintained correctly when the request was executed. This led to inconsistencies that were harmless during normal operation, but these could lead to assertion failures during node failure handling, with subsequent failure of additional nodes. (Bug #22643129)

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

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

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

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

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

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

  References: See also: Bug #22989944.

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

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

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

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

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

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

- Settings for the `SchedulerResponsiveness` data node configuration parameter (introduced in MySQL NDB Cluster 7.4.9) were ignored. (Bug #80341, Bug #22712481)
- When setting CPU spin time, the value was needlessly cast to a boolean internally, so that setting it to any nonzero value yielded an effective value of 1. This issue, as well as the fix for it, apply both to setting the `SchedulerSpinTimer` parameter and to setting `spintime` as part of a `ThreadConfig` parameter value. (Bug #80237, Bug #22647476)
- A logic error in an `if` statement in `storage/ndb/src/kernel/blocks/dbacc/DbaccMain.cpp` rendered useless a check for determining whether `ZREAD_ERROR` should be returned when comparing operations. This was detected when compiling with `gcc` using `-Werror=logical-op`. (Bug #80155, Bug #22601798)

  References: This issue is a regression of: Bug #21285604.
- Builds with the `-Werror` and `-Wextra` flags (as for release builds) failed on SLES 11. (Bug #79950, Bug #22539531)
- When using `CREATE INDEX` to add an index on either of two `NDB` tables sharing circular foreign keys, the query succeeded but a temporary table was left on disk, breaking the foreign key constraints. This issue was also observed when attempting to create an index on a table in the middle of a chain of foreign keys—that is, a table having both parent and child keys, but on different tables. The problem did not occur when using `ALTER TABLE` to perform the same index creation operation; and subsequent analysis revealed unintended differences in the way such operations were performed by `CREATE INDEX`.

  To fix this problem, we now make sure that operations performed by a `CREATE INDEX` statement are always handled internally in the same way and at the same time that the same operations are handled when performed by `ALTER TABLE` or `DROP INDEX`. (Bug #79156, Bug #22173891)
- `NDB` failed to ignore index prefixes on primary and unique keys, causing `CREATE TABLE` and `ALTER TABLE` statements using them to be rejected. (Bug #78441, Bug #21839248)

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

### Bugs Fixed

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

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



### Functionality Added or Changed

- **Important Change:** Previously, the `NDB` scheduler always optimized for speed against throughput in a predetermined manner (this was hard coded); this balance can now be set using the

`SchedulerResponsiveness` data node configuration parameter. This parameter accepts an integer in the range of 0-10 inclusive, with 5 as the default. Higher values provide better response times relative to throughput. Lower values provide increased throughput, but impose longer response times. (Bug #78531, Bug #21889312)

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

## Bugs Fixed

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

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

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

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

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

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

  References: See also: Bug #20646496.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- Functionality Added or Changed
- Bugs Fixed

### Functionality Added or Changed

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

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

    - `BackupDataBufferSize`: minimum increased from 0 to 2M.
    - `BackupLogBufferSize`: minimum increased from 0 to 2M.
    - `BackupWriteSize`: minimum increased from 2K to 32K.
    - `BackupMaxWriteSize`: minimum increased from 2K to 256K.

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

- When a backup was unsuccessful due to insufficient resources, a subsequent retry worked only for those parts of the backup that worked in the same thread, since delayed signals are only supported in the same thread. Delayed signals are no longer sent to other threads in such cases.
- An instance of an internal list object used in searching for queued scans was not actually destroyed before calls to functions that could manipulate the base object used to create it.
- ACC scans were queued in the category of range scans, which could lead to starting an ACC scan when `DBACC` had no free slots for scans. We fix this by implementing a separate queue for ACC scans.

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

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

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

## Bugs Fixed

- **Important Change; NDB Cluster APIs:** The MGM API error-handling functions `ndb_mgm_get_latest_error()`, `ndb_mgm_get_latest_error_msg()`, and `ndb_mgm_get_latest_error_desc()` each failed when used with a `NULL` handle. You should note that, although these functions are now null-safe, values returned in this case are arbitrary and not meaningful. (Bug #78130, Bug #21651706)
- **Important Change:** When `ndb_restore` was run without `--disable-indexes` or `--rebuild-indexes` on a table having a unique index, it was possible for rows to be restored in an order that resulted in duplicate values, causing it to fail with duplicate key errors. Running `ndb_restore` on such a table now requires using at least one of these options; failing to do so now results in an error. (Bug #57782, Bug #11764893)

  References: See also: Bug #22329365, Bug #22345748.
- **NDB Cluster APIs:** While executing `dropEvent()`, if the coordinator `DBDICT` failed after the subscription manager (`SUMA` block) had removed all subscriptions but before the coordinator had deleted the event from the system table, the dropped event remained in the table, causing any subsequent drop or create event with the same name to fail with `NDB` error 1419 `Subscription already dropped` or error 746 `Event name already exists`. This occurred even when calling `dropEvent()` with a nonzero force argument.

  Now in such cases, error 1419 is ignored, and `DBDICT` deletes the event from the table. (Bug #21554676)
- **NDB Cluster APIs:** If the total amount of memory allocated for the event buffer exceeded approximately 40 MB, the calculation of memory usage percentages could overflow during computation. This was due to the fact that the associated routine used 32-bit arithmetic; this has now been changed to use `Uint64` values instead. (Bug #78454, Bug #21847552)
- **NDB Cluster APIs:** The `nextEvent2()` method continued to return exceptional events such as `TE_EMPTY`, `TE_INCONSISTENT`, and `TE_OUT_OF_MEMORY` for event operations which already had been dropped. (Bug #78167, Bug #21673318)
- **NDB Cluster APIs:** After the initial restart of a node following a cluster failure, the cluster failure event added as part of the restart process was deleted when an event that existed prior to the restart was later deleted. This meant that, in such cases, an Event API client had no way of knowing that failure handling was needed. In addition, the GCI used for the final cleanup of deleted event operations, performed by `pollEvents()` and `nextEvent()` when these methods have consumed all available events, was lost. (Bug #78143, Bug #21660947)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  References: See also: Bug #21372136.

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

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

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

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

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

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

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

- The `inet_ntoa()` function used internally in several `mgmd` threads was not `POSIX` thread-safe, which meant that the result it returned could sometimes be undefined. To avoid this problem, a thread-safe and platform-independent wrapper for `inet_ntop()` is used to take the place of this function. (Bug #17766129)

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

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

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

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

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

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

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

- Functionality Added or Changed
- Bugs Fixed

### Functionality Added or Changed

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

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

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

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

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

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

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

- We now ensure that pages created after the LCP has started are not included in LCP scans; we also ensure that no records inserted into those pages have their `LCP_SKIP` flag set.
- Handling of the scan protocol is changed such that a certain amount of progress is made by the LCP regardless of load; we now report progress to the LCP watchdog so that we avoid failure in the event that an LCP is making progress but not writing any records.
- We now take steps to guarantee that LCP scans proceed more quickly than inserts can occur, by ensuring that scans are prioritized this scanning activity, and thus, that the LCP is in fact (eventually) completed.
- In addition, scanning is made more efficient, by prefetching tuples; this helps avoid stalls while fetching memory in the CPU.

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

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

## Bugs Fixed

- **Incompatible Change; NDB Cluster APIs:** The `pollEvents2()` method now returns -1, indicating an error, whenever a negative value is used for the time argument. (Bug #20762291)
- **Important Change; NDB Cluster APIs:** The `Ndb::getHighestQueuedEpoch()` method returned the greatest epoch in the event queue instead of the greatest epoch found after calling `pollEvents2()`. (Bug #20700220)
- **Important Change; NDB Cluster APIs:** `Ndb::pollEvents()` is now compatible with the `TE_EMPTY`, `TE_INCONSISTENT`, and `TE_OUT_OF_MEMORY` event types introduced in MySQL NDB Cluster 7.4.3. For detailed information about this change, see the description of this method in the *MySQL NDB Cluster API Developer Guide*. (Bug #20646496)
- **Important Change; NDB Cluster APIs:** Added the method `Ndb::isExpectingHigherQueuedEpochs()` to the NDB API to detect when additional, newer event epochs were detected by `pollEvents2()`.

  The behavior of `Ndb::pollEvents()` has also been modified such that it now returns **`NDB_FAILURE_GCI`** (equal to `~(Uint64) 0`) when a cluster failure has been detected. (Bug #18753887)
- **NDB Cluster APIs:** Added the `Column::getSizeInBytesForRecord()` method, which returns the size required for a column by an `NdbRecord`, depending on the column's type (text/blob, or other). (Bug #21067283)
- **NDB Cluster APIs:** `NdbEventOperation::isErrorEpoch()` incorrectly returned `false` for the `TE_INCONSISTENT` table event type (see Event::TableEvent). This caused a subsequent call to `getEventType()` to fail. (Bug #20729091)
- **NDB Cluster APIs:** Creation and destruction of `Ndb_cluster_connection` objects by multiple threads could make use of the same application lock, which in some cases led to failures in the global dictionary cache. To alleviate this problem, the creation and destruction of several internal NDB API objects have been serialized. (Bug #20636124)
- **NDB Cluster APIs:** A number of timeouts were not handled correctly in the NDB API. (Bug #20617891)
- **NDB Cluster APIs:** When an `Ndb` object created prior to a failure of the cluster was reused, the event queue of this object could still contain data node events originating from before the failure. These events could reference “old” epochs (from before the failure occurred), which in turn could

violate the assumption made by the `nextEvent()` method that epoch numbers always increase. This issue is addressed by explicitly clearing the event queue in such cases. (Bug #18411034)

References: See also: Bug #20888668.

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

  References: See also: Bug #16890703.

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

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

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

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

  References: See also: Bug #20481140.

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

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

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

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

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

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

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

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

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

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

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

  References: See also: Bug #17916243.

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

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

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

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

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

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

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

  (Bug #77081, Bug #21109605)

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

## Bugs Fixed

- During backup, loading data from one SQL node followed by repeated `DELETE` statements on the tables just loaded from a different SQL node could lead to data node failures. (Bug #18949230)
- When an instance of `NdbEventBuffer` was destroyed, any references to GCI operations that remained in the event buffer data list were not freed. Now these are freed, and items from the event bufer data list are returned to the free list when purging GCI containers. (Bug #76165, Bug #20651661)
- When a bulk delete operation was committed early to avoid an additional round trip, while also returning the number of affected rows, but failed with a timeout error, an SQL node performed no verification that the transaction was in the Committed state. (Bug #74494, Bug #20092754)

  References: See also: Bug #19873609.

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

## Bugs Fixed

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

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

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

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

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

  - Result rows were unpacked to full `NdbRecord` format before they were stored in the buffer. If only some but not all columns of a table were selected, the buffer contained empty space (essentially wasted).
  - Due to the buffer format being unpacked, `VARCHAR` and `VARBINARY` columns always had to be allocated for the maximum size defined for such columns.
  - `BatchByteSize` and `MaxScanBatchSize` values were not taken into consideration as a limiting factor when calculating the maximum buffer size.

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

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

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

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

  References: See also: Bug #34702.

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

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

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

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

  References: See also: Bug #44529.

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

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

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

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

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

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

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

## Bugs Fixed

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

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

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

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

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

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

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

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

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

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

- The `ndbinfo.restart_info` table did not contain a new row as expected following a node restart. (Bug #75825, Bug #20504971)
- The output format of `SHOW CREATE TABLE` for an `NDB` table containing foreign key constraints did not match that for the equivalent `InnoDB` table, which could lead to issues with some third-party applications. (Bug #75515, Bug #20364309)
- An `ALTER TABLE` statement containing comments and a partitioning option against an `NDB` table caused the SQL node on which it was executed to fail. (Bug #74022, Bug #19667566)

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

- Functionality Added or Changed
- Bugs Fixed

### Functionality Added or Changed

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

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

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

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

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

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

- **NDB Cluster APIs:** Two new example programs, demonstrating reads and writes of `CHAR`, `VARCHAR`, and `VARBINARY` column values, have been added to `storage/ndb/ndbapi-examples` in the MySQL NDB Cluster source tree. For more information about these programs, including source code listings, see NDB API Simple Array Example, and NDB API Simple Array Example Using Adapter.
- Additional logging is now performed of internal states occurring during system restarts such as waiting for node ID allocation and master takeover of global and local checkpoints. (Bug #74316, Bug #19795029)

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

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

## Bugs Fixed

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

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

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

  References: See also: Bug #19846392.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- Functionality Added or Changed

- Bugs Fixed

### Functionality Added or Changed

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

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

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

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

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

### Bugs Fixed

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

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

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

- The fix for a previous issue with the handling of multiple node failures required determining the number of TC instances the failed node was running, then taking them over. The mechanism to determine this number sometimes provided an invalid result which caused the number of TC instances in the failed node to be set to an excessively high value. This in turn caused redundant

takeover attempts, which wasted time and had a negative impact on the processing of other node failures and of global checkpoints. (Bug #19193927)

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

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

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

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

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

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

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

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



### Node Restart Performance and Reporting Enhancements

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

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

  - When performing a node or system restart, it is necessary to restore local checkpoints for the fragments. This process previously used delayed signals at a point which was found to be critical

to performance; these have now been replaced with normal (undelayed) signals, which should shorten significantly the time required to back up a MySQL NDB Cluster or to restore it from backup.

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

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

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

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

## Improved Scan and SQL Processing

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

## Per-Fragment Memory Reporting

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

## Bugs Fixed

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

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

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

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

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

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

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

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

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

# Index

## Symbols

## A



## B



## C

## D

## E

## F



## G



## H



## I

## J



## K



## L



## M

## N

## O



## P

## Q



## R



## S

## T



## U

## V



## W



## X



## Z