MySQL NDB Cluster 7.6 Release Notes

Abstract

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

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

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

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

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

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

For legal information, see the Legal Notices.

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

Document generated on: 2026-04-28 (revision: 31150)

Table of Contents

Preface and Legal Notices 3
Changes in MySQL NDB Cluster 7.6.38 (5.7.44-ndb-7.6.38) (2026-04-23) 4
Changes in MySQL NDB Cluster 7.6.37 (5.7.44-ndb-7.6.37) (2026-01-22) 5
Changes in MySQL NDB Cluster 7.6.36 (5.7.44-ndb-7.6.36) (2025-10-23) 5
Changes in MySQL NDB Cluster 7.6.35 (5.7.44-ndb-7.6.35) (2025-07-23) 6
Changes in MySQL NDB Cluster 7.6.34 (5.7.44-ndb-7.6.34) (2025-04-16) 7
Changes in MySQL NDB Cluster 7.6.33 (5.7.44-ndb-7.6.33) (2025-01-22) 8
Changes in MySQL NDB Cluster 7.6.32 (5.7.44-ndb-7.6.32) (2024-10-15) 10
Changes in MySQL NDB Cluster 7.6.31 (5.7.44-ndb-7.6.31) (2024-07-02) 11
Changes in MySQL NDB Cluster 7.6.30 (5.7.44-ndb-7.6.30) (2024-04-30) 12
Changes in MySQL NDB Cluster 7.6.29 (5.7.44-ndb-7.6.29) (2024-01-16) 13
Changes in MySQL NDB Cluster 7.6.28 (5.7.44-ndb-7.6.28) (2023-10-26) 15
Changes in MySQL NDB Cluster 7.6.27 (5.7.43-ndb-7.6.27) (2023-07-18) 17
Changes in MySQL NDB Cluster 7.6.26 (5.7.42-ndb-7.6.26) (2023-04-19) 19
Changes in MySQL NDB Cluster 7.6.25 (5.7.41-ndb-7.6.25) (2023-01-18) 22
Changes in MySQL NDB Cluster 7.6.24 (5.7.40-ndb-7.6.24) (2202-10-12) 24

Changes in MySQL NDB Cluster 7.6.23 (5.7.39-ndb-7.6.23) (2022-07-27) 25
Changes in MySQL NDB Cluster 7.6.22 (5.7.38-ndb-7.6.22) (2022-04-27) 26
Changes in MySQL NDB Cluster 7.6.21 (5.7.37-ndb-7.6.21) (2022-01-19) 27
Changes in MySQL NDB Cluster 7.6.20 (5.7.36-ndb-7.6.20) (2021-10-20) 30
Changes in MySQL NDB Cluster 7.6.19 (5.7.35-ndb-7.6.19) (2021-07-21) 31
Changes in MySQL NDB Cluster 7.6.18 (5.7.34-ndb-7.6.18) (2021-04-21) 32
Changes in MySQL NDB Cluster 7.6.17 (5.7.33-ndb-7.6.17) (2021-01-19) 33
Changes in MySQL NDB Cluster 7.6.16 (5.7.32-ndb-7.6.16) (2020-10-20) 35
Changes in MySQL NDB Cluster 7.6.15 (5.7.31-ndb-7.6.15) (2020-07-14) 36
Changes in MySQL NDB Cluster 7.6.14 (5.7.30-ndb-7.6.14) (2020-04-28) 38
Changes in MySQL NDB Cluster 7.6.13 (5.7.29-ndb-7.6.13) (2020-01-14) 41
Changes in MySQL NDB Cluster 7.6.12 (5.7.28-ndb-7.6.12) (2019-10-15) 44
Changes in MySQL NDB Cluster 7.6.11 (5.7.27-ndb-7.6.11) (2019-07-23) 46
Changes in MySQL NDB Cluster 7.6.10 (5.7.26-ndb-7.6.10) (2019-04-26) 48
Changes in MySQL NDB Cluster 7.6.9 (5.7.25-ndb-7.6.9) (2019-01-22) 51
Changes in MySQL NDB Cluster 7.6.8 (5.7.24-ndb-7.6.8) (2018-10-23) 53
Changes in MySQL NDB Cluster 7.6.7 (5.7.23-ndb-7.6.7) (2018-07-27) 56
Changes in MySQL NDB Cluster 7.6.6 (5.7.22-ndb-7.6.6) (2018-05-31) 58
Changes in MySQL NDB Cluster 7.6.5 (5.7.20-ndb-7.6.5) (2018-04-20, Development) 62
Changes in MySQL NDB Cluster 7.6.4 (5.7.20-ndb-7.6.4) (2018-01-31, Development Milestone 4) 63
Changes in MySQL NDB Cluster 7.6.3 (5.7.18-ndb-7.6.3) (2017-07-03, Development Milestone 3) 71
Changes in MySQL NDB Cluster 7.6.2 (5.7.18-ndb-7.6.2) (2017-04-26, Development Milestone 2) 76
Changes in MySQL NDB Cluster 7.6.1 (5.7.17-ndb-7.6.1) (Not released, Development Milestone 1) 81
Release Series Changelogs: MySQL NDB Cluster 7.6 83
Changes in MySQL NDB Cluster 7.6.30 (5.7.44-ndb-7.6.30) (2024-04-30) 84
Changes in MySQL NDB Cluster 7.6.29 (5.7.44-ndb-7.6.29) (2024-01-16) 84
Changes in MySQL NDB Cluster 7.6.28 (5.7.44-ndb-7.6.28) (2023-10-26) 86
Changes in MySQL NDB Cluster 7.6.27 (5.7.43-ndb-7.6.27) (2023-07-18) 87
Changes in MySQL NDB Cluster 7.6.26 (5.7.42-ndb-7.6.26) (2023-04-19) 89
Changes in MySQL NDB Cluster 7.6.25 (5.7.41-ndb-7.6.25) (2023-01-18) 91
Changes in MySQL NDB Cluster 7.6.24 (5.7.40-ndb-7.6.24) (2202-10-12) 93
Changes in MySQL NDB Cluster 7.6.23 (5.7.39-ndb-7.6.23) (2022-07-27) 93
Changes in MySQL NDB Cluster 7.6.22 (5.7.38-ndb-7.6.22) (2022-04-27) 94
Changes in MySQL NDB Cluster 7.6.21 (5.7.37-ndb-7.6.21) (2022-01-19) 95
Changes in MySQL NDB Cluster 7.6.20 (5.7.36-ndb-7.6.20) (2021-10-20) 97
Changes in MySQL NDB Cluster 7.6.19 (5.7.35-ndb-7.6.19) (2021-07-21) 98
Changes in MySQL NDB Cluster 7.6.18 (5.7.34-ndb-7.6.18) (2021-04-21) 99
Changes in MySQL NDB Cluster 7.6.17 (5.7.33-ndb-7.6.17) (2021-01-19) 100
Changes in MySQL NDB Cluster 7.6.16 (5.7.32-ndb-7.6.16) (2020-10-20) 101
Changes in MySQL NDB Cluster 7.6.15 (5.7.31-ndb-7.6.15) (2020-07-14) 102
Changes in MySQL NDB Cluster 7.6.14 (5.7.30-ndb-7.6.14) (2020-04-28) 104
Changes in MySQL NDB Cluster 7.6.13 (5.7.29-ndb-7.6.13) (2020-01-14) 107
Changes in MySQL NDB Cluster 7.6.12 (5.7.28-ndb-7.6.12) (2019-10-15) 110
Changes in MySQL NDB Cluster 7.6.11 (5.7.27-ndb-7.6.11) (2019-07-23) 112
Changes in MySQL NDB Cluster 7.6.10 (5.7.26-ndb-7.6.10) (2019-04-26) 113
Changes in MySQL NDB Cluster 7.6.9 (5.7.25-ndb-7.6.9) (2019-01-22) 116
Changes in MySQL NDB Cluster 7.6.8 (5.7.24-ndb-7.6.8) (2018-10-23) 118
Changes in MySQL NDB Cluster 7.6.7 (5.7.23-ndb-7.6.7) (2018-07-27) 121
Changes in MySQL NDB Cluster 7.6.6 (5.7.22-ndb-7.6.6) (2018-05-31) 122
Changes in MySQL NDB Cluster 7.6.5 (5.7.20-ndb-7.6.5) (2018-04-20, Development) 126
Changes in MySQL NDB Cluster 7.6.4 (5.7.20-ndb-7.6.4) (2018-01-31, Development Milestone 4) 127
Changes in MySQL NDB Cluster 7.6.3 (5.7.18-ndb-7.6.3) (2017-07-03, Development Milestone 3) 134

Changes in MySQL NDB Cluster 7.6.2 (5.7.18-ndb-7.6.2) (2017-04-26, Development Milestone 2) 138
Changes in MySQL NDB Cluster 7.6.1 (5.7.17-ndb-7.6.1) (Not released, Development Milestone 1) 143
Index 145

Preface and Legal Notices

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

Legal Notices

Copyright © 1997, 2026, Oracle and/or its affiliates.

License Restrictions

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

Warranty Disclaimer

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

Restricted Rights Notice

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

Hazardous Applications Notice

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Trademark Notice

Oracle, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

Third-Party Content, Products, and Services Disclaimer

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Use of This Documentation

This documentation is NOT distributed under a GPL license. Use of this documentation is subject to the following terms:

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how Oracle disseminates it (that is, electronically for download on a Web site with the software) or on a CD-ROM or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Oracle. Oracle and/or its affiliates reserve any and all rights to this documentation not expressly granted above.

Documentation Accessibility

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

Access to Oracle Support for Accessibility

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

Changes in MySQL NDB Cluster 7.6.38 (5.7.44-ndb-7.6.38) (2026-04-23)

MySQL NDB Cluster 7.6.38 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.44 (see Changes in MySQL 5.7.44 (2023-10-25)).

Version 5.7.44-ndb-7.6.38 has no release notes, or they have not been published because the product version has not been released.

Changes in MySQL NDB Cluster 7.6.37 (5.7.44-ndb-7.6.37) (2026-01-22)

MySQL NDB Cluster 7.6.37 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.44 (see Changes in MySQL 5.7.44 (2023-10-25)).

InnoDB Notes

- Fixed an issue related to bulk inserts. (Bug #38208188)

Changes in MySQL NDB Cluster 7.6.36 (5.7.44-ndb-7.6.36) (2025-10-23)

MySQL NDB Cluster 7.6.36 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.44 (see Changes in MySQL 5.7.44 (2023-10-25)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- MySQL NDB Cluster now generates log entries for redo logging issues, including `redo log buffer exhaustion`, `redo log space exhaustion`, and `file change problems`, which can cause transactions to be aborted, queued, or delayed. A secondary overload control mechanism monitors

the rate at which the redo log part's Redo buffer is flushed to disk and estimates the time required to complete writing all data in the part's Redo buffer. (Bug #37903091)

Bugs Fixed

- It was not possible to restore backups with `BackupID` values greater than 2147483647, with the `ndb_restore` tool. Errors were returned similar to the following:

```
ndb_restore: [Warning] option 'backupid': signed value
3000000000 adjusted to 2147483647. Failed to find backup
2147483647.
```

(Bug #38260769)

Changes in MySQL NDB Cluster 7.6.35 (5.7.44-ndb-7.6.35) (2025-07-23)

MySQL NDB Cluster 7.6.35 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.44 (see Changes in MySQL 5.7.44 (2023-10-25)).

- Compilation Notes
- Functionality Added or Changed
- Bugs Fixed

Compilation Notes

- **Solaris:** Clang and GCC now can be used for compiling MySQL on Solaris. (Bug #30562248)
- OpenSSL 3 is now used on Solaris. (Bug #38193362)
- MySQL Server now supports CMake 4, ensuring compatibility with future CMake versions where support for versions prior to 3.10 is expected to be discontinued. (Bug #38027636)

Functionality Added or Changed

- **Important Change:** Added the `mysql` client `--commands` option, which enables or disables most `mysql` client commands.

  This option is enabled by default. To disable it, start the client with `--commands=OFF` or `--skip-commands`.

  For a complete list of all commands affected by this option, and additional information, see mysql Client Options. (WL #16949)

  References: See also: Bug #36416568, Bug #38066040.

Bugs Fixed

- **InnoDB:** Fixed an issue relating to range queries on tables. (Bug #31360522)

  References: See also: Bug #38063122.

- Following an upgrade from NDB 8.0 to NDB 8.4, all data nodes in the cluster underwent an unexpected simultaneous restart. This occurred when the transaction coordinator had no scan state, leading to protocol timeout; the resulting misalignment in protocol states caused data nodes to shut down unexpectedly. This is fixed by extending existing handling of an unexpected `SCAN_NEXTREQ` signal to cover the case when the scan is already stateless. (Bug #37994985)

  References: This issue is a regression of: Bug #37022901.

- `WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS()` did not always execute correctly. (Bug #37829550)

  References: See also: Bug #38063175.

- Unquoted semicolon characters (`;`) within comments were not always flagged as errors, in spite of the fact that they are not allowed. (Bug #37117875)

  References: See also: Bug #38063286.

Changes in MySQL NDB Cluster 7.6.34 (5.7.44-ndb-7.6.34) (2025-04-16)

MySQL NDB Cluster 7.6.34 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.44 (see Changes in MySQL 5.7.44 (2023-10-25)).

Bugs Fixed

- **InnoDB:** Fixed an issue relating to reading `index_id` values. (Bug #36993445, Bug #37709706)

- **Replication:** In some cases, `MASTER_POS_WAIT()` did not perform as expected. (Bug #36421684, Bug #37709187)

- `mysqldump` did not escape certain special characters properly in its output. With this fix, `mysqldump` now follows the rules as described in String Literals. (Bug #37540722, Bug #37709163)

- API node failure is detected by one or more data nodes; data nodes detecting API node failure inform all other data nodes of the failure, eventually triggering API node failure handling on each data node.

  Each data node handles API node failure independently; once all internal blocks have completed cleanup, the API node failure is considered handled, and, after a timed delay, the `QMGR` block allows the failed API node's node ID to be used for new connections.

`QMGR` monitors API node failure handling, periodically generating warning logs for API node failure handling that has not completed (approximately every 30 seconds). These logs indicate which blocks have yet to complete failure handling.

This enhancement improves logging in handling stalls particularly with regard to the `DBTC` block, which must roll back or commit and complete the API node's transactions, and release the associated `COMMIT` and `ACK` markers. In addition, the time to wait for API node failure handling is now configurable as the `ApiFailureHandlingTimeout` data node configuration parameter; after this number of seconds, handling is escalated to a data node restart. (Bug #37524092)

References: See also: Bug #37469364.

- When a data node hangs during shutdown reasons for this may include: I/O problems on the node, in which case the thread shutting down hangs while operating on error and trace files; or an error in the shutdown logic, where the thread shutting down raises a Unix signal, and causes a deadlock. When such issues occur, users might observe watchdog warnings in the logs, referring to the last signal processed; this could be misleading in cases where there was actually a (different) preceding cause which had triggered the shutdown.

  To help pinpoint the origin of such problems if they occur, we have made the following improvements:

  - Added a new watchdog state `shutting down`. This is set early enough in the error handling process that it causes all watchdog logging of shutdown stalls to attribute the delay to a shutdown delay (correctly) rather than problem in execution.
  - We have also modified the watchdog mechanism to be aware of shutdown states, and use a more direct path—which is less likely to stall—to force the data node process to stop when needed.

  (Bug #37518267)

- Signal dump code run when handling an unplanned node shutdown sometimes exited unexpectedly when speculatively reading section IDs which might not be present. (Bug #37512526)
- The LQH_TRANSCONF signal printer did not validate its input length correctly, which could lead the node process to exit. (Bug #37512477)
- Removed an issue relating to invalid UTF8 values. (Bug #27618273, Bug #37709687)
- Addressed an issue relating to an invalid identifier. (Bug #22958632, Bug #37709664)

Changes in MySQL NDB Cluster 7.6.33 (5.7.44-ndb-7.6.33) (2025-01-22)

MySQL NDB Cluster 7.6.33 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.44 (see Changes in MySQL 5.7.44 (2023-10-25)).

- Compilation Notes
- Bugs Fixed

Compilation Notes

- Added `CONTRIBUTING.md` and `SECURITY.md` files to the MySQL sources to conform to Oracle's Open Source guidelines. (Bug #36998165)

Bugs Fixed

- **InnoDB:** An assertion failure was raised when creating a `FULLTEXT` index on a table with an `FTS_DOC_ID` value greater than 4294967295. (Bug #36879147)

  References: See also: Bug #37387224.

- **NDB Cluster APIs:** Removed known causes of API node versus data node state misalignments, and improved the handling of state misalignments when detected. In one such case, separate handling of scan errors in the `NDB` kernel and those originating in API programs led to cleanup not being performed after some scans. Handling of `DBTC` and API state alignment errors has been improved by this set of fixes, as well as scan protocol timeout handling in `DBSPJ`; now, when such misalignments in state are detected, the involved API nodes are disconnected rather than the data node detecting it being forced to shut down. (Bug #20430083, Bug #22782511, Bug #23528433, Bug #28505289, Bug #36273474, Bug #36395384, Bug #36838756, Bug #37022773, Bug #37022901, Bug #37023549)

  References: See also: Bug #22782511, Bug #23528433, Bug #36273474, Bug #36395384, Bug #36838756.

- **ndbinfo Information Database:** At table create and drop time, access of `ndbinfo` tables such as `operations_per_fragment` and `memory_per_fragment` sometimes examined data which was not valid.

  To fix this, during scans of these `ndbinfo` tables, we ignore any fragments from tables in transient states at such times due to being created or dropped. (Bug #37140331)

- In some cases, the occurrence of node failures during shutdown led to the cluster becoming unrecoverable without manual intervention.

  We fix this by modifying global checkpoint ID (GCI) information propagation (`CopyGCI` mechanism) to reject propagation of any set of GCI information which does not describe the ability to recover the cluster automatically as part of a system restart. (Bug #37163647)

  References: See also: Bug #37162636.

- In some cases, node failures during an otherwise graceful shutdown could lead to a cluster becoming unrecoverable without manual intervention. This fix modifies the generic GCI info propagation mechanism (`CopyGCI`) to reject propagating any set of GCI information which does not describe the ability to recover a cluster automatically. (Bug #37162636)

- AppArmor denied access to `/proc/$pid/task/$thread_id/mem`, a file required to generate a stack trace. (Bug #37063288)

  References: See also: Bug #37387034.

- In cases where `NDB` experienced an API protocol timeout when attempting to close a scan operation, it considered the `DBTC` `ApiConnectRecord` involved to be lost for further use, at least until the API disconnected and API failure handling within `DBTC` reclaimed the record.

This has been improved by having the API send a `TCRELEASEREQ` signal to `DBTC` in such cases, performing API failure handling for a single `ApiConnectRecord` within `DBTC`. (Bug #37023661)

References: See also: Bug #36273474, Bug #36395384, Bug #37022773, Bug #37022901, Bug #37023549.

- Improved the internal function `my_print_help()`. (Bug #36615714)

  References: See also: Bug #37387224.

- A subquery containing an aggregate function `WITH ROLLUP` which was part of a row value comparator was not always processed correctly. (Bug #36593235)

  References: See also: Bug #37387180. This issue is a regression of: Bug #30969045, Bug #30921780, Bug #26227613, Bug #29134467, Bug #30967158.

- Testing revealed that a fix for a previous issue which added a check of the `ApiConnectRecord` failure number against the system's current failure number did not initialize the `ApiConnectRecord` failure number in all cases. (Bug #36155195)

  References: This issue is a regression of: Bug #36028828.

Changes in MySQL NDB Cluster 7.6.32 (5.7.44-ndb-7.6.32) (2024-10-15)

MySQL NDB Cluster 7.6.32 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.44 (see Changes in MySQL 5.7.44 (2023-10-25)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change:** For platforms on which OpenSSL libraries are bundled, the linked OpenSSL library for MySQL Server has been updated to version 3.0.15. For more information, see OpenSSL 3.0 Series Release Notes and OpenSSL Security Advisory [3rd September 2024]. (Bug #37021075)

Bugs Fixed

- **NDB Cluster APIs:** Using `NdbRecord` and `OO_SETVALUE` from the NDB API to write the value of a `Varchar`, `Varbinary`, `Longvarchar`, or `Longvarbinary` column failed with error `829`. (Bug #36989337)
- **MySQL NDB ClusterJ:** `ReconnectTest` in the ClusterJ test suite failed sometimes due to a race condition. The test has been rewritten with proper synchronization. (Bug #28550140)

- Fixed an issue relating to FTS comparisons.

  Our thanks to Shaohua Wang and the team at Alibaba for the contribution. (Bug #37039409)

- While dumping tablespaces, `mysqldump` did not properly escape certain SQL statements in its output. In addition, the dump now encloses the following identifiers within backticks: `LOGFILE GROUP`, `TABLESPACE`, and `ENGINE`. (Bug #37039394)

- The `AES_ENCRYPT()` function did not always return a valid result. (Bug #37039383)

- Removed node management code from `TRIX` that was not actually used. (Bug #37006547)

- Submitting concurrent shutdown commands for individual nodes using `ndb_mgm SHUTDOWN` *`node_id`* or the MGM API sometimes had one or both of the following adverse results:

  - Cluster failure when all nodes in the same node group were stopped
  - Inability to recover when all nodes in the same node group were stopped, and the cluster had more than one node group

  This was due to the fact that the (planned) shutdown of a single node assumed that only one such shutdown occurred at a time, but did not actually check this limitation.

  We fix this so that concurrent single-node shutdown requests are serialized across the cluster, and any which would cause a cluster outage are rejected. (Bug #36943756)

  References: See also: Bug #36839995.

- Shutdown of a data node late in a schema transaction updating index statistics caused the president node to shut down as well. (Bug #36886242)

  References: See also: Bug #36877952.

- It was possible for duplicate events to be sent to user applications when a data node was shut down. (Bug #36750146)

- The server did not always handle connections correctly when running with both the thread pool and audit log plugins. (Bug #36682079)

- Issues arose when an attempt was made to use a SHM transporter's wakeup socket before it was ready, due in part to error-handling when setting up the SHM transporter, which did not close the socket correctly prior to making another attempt at setup. (Bug #36568752, Bug #36623058)

- `DROP INDEX` with the addition of a `FULLTEXT` index in the same transaction sometimes led to an unplanned server exit. (Bug #36559642)

Changes in MySQL NDB Cluster 7.6.31 (5.7.44-ndb-7.6.31) (2024-07-02)

MySQL NDB Cluster 7.6.31 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.44 (see Changes in MySQL 5.7.44 (2023-10-25)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **NDB Client Programs:** When started, `ndbd` now produces a warning in the data node log like this one:

  ```
  2024-05-28 13:32:16 [ndbd] WARNING  -- Running ndbd with a single thread of
  signal execution.  For multi-threaded signal execution run the ndbmtd binary.
  ```

  (Bug #36326896)

Bugs Fixed

- **InnoDB:** Some FTS operations on tables with FTS indexes led to inconsistent results. For example, if the server terminated while synchronizing the FTS cache or when synchronization occurred concurrently with another FTS operation.

  Our thanks to Yin Peng and the Tencent team for the contribution. (Bug #36347647)

- **NDB Client Programs:** `ndb_restore` did not restore a foreign key whose columns differed in order from those of the parent key.

  Our thanks to Axel Svensson for the contribution. (Bug #114147, Bug #36345882)

- Averages of certain numbers were not always computed correctly. (Bug #36741923)

  References: See also: Bug #36563773.

- Running two concurrent `OPTIMIZE TABLE` statements on the same table with fulltext indexes and `innodb_optimize_fulltext_only` enabled sometimes caused the server to exit. (Bug #36741880)

  References: See also: Bug #36741880.

- When distribution awareness was not in use, the cluster tended to choose the same data node as the transaction coordinator repeatedly. (Bug #35840020, Bug #36554026)

Changes in MySQL NDB Cluster 7.6.30 (5.7.44-ndb-7.6.30) (2024-04-30)

MySQL NDB Cluster 7.6.30 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.44 (see Changes in MySQL 5.7.44 (2023-10-25)).

Bugs Fixed

- **NDB Client Programs:** `ndb_redo_log_reader` exited with `Record type = 0 not implemented` when reaching an unused page, all zero bytes, or a page which was only partially used (typically a page consisting of the page header only). (Bug #36313259)
- `SET GLOBAL offline_mode=ON` did not always perform correctly when issued under high loads. (Bug #36275182)

  References: See also: Bug #36405894.
- `mysqldump` did not always interpret the server version correctly. (Bug #36248967)

  References: See also: Bug #36405879.
- Repeated incomplete incomplete attempts to perform a system restart in some cases left the cluster in a state from which it could not recover without restoring it from backup. (Bug #35801548)
- The event buffer used by the NDB API maintains an internal pool of free memory to reduce the interactions with the runtime and operating system, while allowing memory that is no longer needed to be returned for other uses. This free memory is subtracted from the total allocated memory to determine the memory is use which is reported and used for enforcing buffer limits and other purposes; this was represented using a 32-bit value, so that if it exceeded 4 GB, the value wrapped, and the amount of free memory appeared to be reduced. This had potentially adverse effects on event buffer memory release to the runtime and OS, free memory reporting, and memory limit handling.

  This is fixed by using a 64-bit value to represent the amount of pooled free memory. (Bug #35483764)

  References: See also: Bug #35655162, Bug #35663761.
- Removed unnecessary warnings generated by transient disconnections of data nodes during restore operations. (Bug #33144487)
- In some cases, when trying to perform an online add index operation on an `NDB` table with no explicit primary key (see Limitations of NDB online operations), the resulting error message did not make the nature of the problem clear. (Bug #30766579)

  References: See also: Bug #36382071.
- Removed an assertion failure in `sql/field.cc`. (Bug #112503, Bug #35846221)

Changes in MySQL NDB Cluster 7.6.29 (5.7.44-ndb-7.6.29) (2024-01-16)

MySQL NDB Cluster 7.6.29 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.44 (see Changes in MySQL 5.7.44 (2023-10-25)).

- Compilation Notes
- Pluggable Authentication
- Bugs Fixed

Compilation Notes

- **Microsoft Windows:** NDB Cluster did not compile correctly using Visual Studio 2022. (Bug #35967676)
- NDB Cluster did not compile correctly on Ubuntu 23.10. (Bug #35847193)

Pluggable Authentication

- Beginning with this release, the behavior of the `AUTHENTICATION_PAM_LOG` environment variable used in debugging the PAM authentication plugin is changed as follows:

  - Setting `AUTHENTICATION_PAM_LOG` to an arbitrary value (except as noted in the next item) no longer includes passwords in its diagnostic messages.
  - To include passwords in the diagnostic messages, set `AUTHENTICATION_PAM_LOG=PAM_LOG_WITH_SECRET_INFO`.

  For more information, see PAM Authentication Debugging. (Bug #74313, Bug #20042010)

Bugs Fixed

- When a node failure is detected, transaction coordinator (TC) instances check their own transactions to determine whether they need handling to ensure completion, implemented by checking whether each transaction involves the failed node, and if so, marking it for immediate timeout handling. This causes the transaction to be either rolled forward (commit) or back (abort), depending on whether it had started committing, using the serial commit protocol. When the TC was in the process of getting permission to commit (`CS_PREPARE_TO_COMMIT`), sending commit requests (`CS_COMMITTING`), or sending completion requests (`CS_COMPLETING`), timeout handling waited until the transaction was in a stable state before commencing the serial commit protocol.

  Prior to the fix for Bug#22602898, all timeouts during `CS_COMPLETING` or `CS_COMMITTING` resulted in switching to the serial commit-complete protocol, so skipping the handling in any of the three states cited previously did not stop the prompt handling of the node failure. It was found later that this fix removed the blanket use of the serial commit-complete protocol for commit-complete timeouts, so that when handling for these states was skipped, no node failure handling action was taken, with the result that such transactions hung in a commit or complete phase, blocking checkpoints.

  The fix for Bug#22602898 removed this stable state handling to avoid it accidentally triggering, but this change also stopped it from triggering when needed in this case where node failure handling found a transaction in a transient state. We solve this problem by modifying `CS_COMMIT_SENT` and `CS_COMPLETE_SENT` stable state handling to perform node failure processing if a timeout has occurred for a transaction with a failure number different from the current latest failure number, ensuring that all transactions involving the failed node are in fact eventually handled. (Bug #36028828)

  References: See also: Bug #22602898.

- It was possible for the `readln_socket()` function in `storage/ndb/src/common/util/socket_io.cpp` to read one character too many from the buffer passed to it as an argument. (Bug #35857936)

- In limited cases, passing data to the `MD5()` encryption function could halt the server. (Bug #35764496)
- The slow disconnection of a data node while a management server was unavailable could sometimes interfere with the rolling restart process. This became especially apparent when the cluster was hosted by NDB Operator, and the old `mgmd` pod did not recognize the IP address change of the restarted data node pod; this was visible as discrepancies in the output of `SHOW STATUS` on different management nodes.

  We fix this by making sure to clear any cached address when connecting to a data node so that the data node's new address (if any) is used instead. (Bug #35667611)
- The maximum permissible value for the oldest restorable global checkpoint ID is `MAX_INT32` (4294967295). Such an ID greater than this value causes the data node to shut down, requiring a backup and restore on a cluster started with `--initial`.

  Now, approximately 90 days before this limit is reached under normal usage, an appropriate warning is issued, allowing time to plan the required corrective action. (Bug #35641420)

  References: See also: Bug #35749589.
- Subscription reports were sent out too early by `SUMA` during a node restart, which could lead to schema inconsistencies between cluster SQL nodes. In addition, an issue with the `ndbinfo restart_info` table meant that restart phases for nodes that did not belong to any node group were not always reported correctly. (Bug #30930132)
- Online table reorganization inserts rows from existing table fragments into new table fragments; then, after committing the inserted rows, it deletes the original rows. It was found that the inserts caused `SUMA` triggers to fire, and binary logging to occur, which led to the following issues:
  - Inconsistent behavior, since DDL is generally logged as one or more statements, if at all, rather than by row-level effect.
  - It was incorrect, since only writes were logged, but not deletes.
  - It was unsafe since tables with blobs did not receive associated the row changes required to form valid binary log events.
  - It used CPU and other resources needlessly.

  For tables with no blob columns, this was primarily a performance issue; for tables having blob columns, it was possible for this behavior to result in unplanned shutdowns of mysqld processes performing binary logging and perhaps even data corruption downstream. (Bug #19912988)

  References: See also: Bug #16028096, Bug #34843617.

Changes in MySQL NDB Cluster 7.6.28 (5.7.44-ndb-7.6.28) (2023-10-26)

MySQL NDB Cluster 7.6.28 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.44 (see Changes in MySQL 5.7.44 (2023-10-25)).

Bugs Fixed

- **InnoDB:** The last detected deadlock section of the engine status log was only showing 1024 characters for the combined thread and query information. Fixing by removing the printed query string limit. (Bug #80927, Bug #23036096)

- **NDB Replication:** Updates to primary keys of character types were not correctly represented in the `BEFORE` and `AFTER` trigger values sent to the NDB binary log injector. This issue was previously fixed in part, but it was discovered subsequently that the problem still occurred when the `mysqld` was run with the binary logging options having the values listed here:

  - `--ndb-log-update-minimal=ON`
  - `--ndb-log-update-as-write=OFF`

  The minimal binary log format excluded all primary key columns from the `AFTER` values reflecting the updated row, the rationale for this being a flawed assumption that the primary key remained constant when an update trigger was received. This did not take into account the fact that, if the primary key uses a character data type, an update trigger is received if character columns are updated to values treated as equal by the comparison rules of the collation used.

  To be able to replicate such changes, we need to include them in the `AFTER` values; this fix ensures that we do so. (Bug #34540016)

  References: See also: Bug #27522732, Bug #34312769, Bug #34388068.

- **NDB Cluster APIs:** `Ndb::pollEvents2()` did not set **`NDB_FAILURE_GCI`** (`~(Uint64)0`) to indicate cluster failure. (Bug #35671818)

  References: See also: Bug #31926584. This issue is a regression of: Bug #18753887.

- **NDB Client Programs:** When `ndb_select_all` failed to read all data from the table, it always tried to re-read it. This could lead to the two problems listed here:

  - Returning a non-empty partial result eventually led to spurious reports of duplicate rows.
  - The table header was printed on every retry.

  Now when `ndb_select_all` is unsuccessful at reading all the table data, its behavior is as follows:

  - When the result is non-empty, `ndb_select_all` halts with an error (and does not retry the scan of the table).
  - When the result is empty, `ndb_select_all` retries the scan, reusing the old header.

  (Bug #35510814)

- Following a node connection failure, the transporter registry's error state was not cleared before initiating a reconnect, which meant that the error causing the connection to be disconnected originally might still be set; this was interpreted as a failure to reconnect. (Bug #35774109)

- When a `TransporterRegistry` (TR) instance connects to a management server, it first uses the MGM API, and then converts the connection to a `Transporter` connection for further communication.

The initial connection had an excessively long timeout (60 seconds) so that, in the case of a cluster having two management servers where one was unavailable, clients were forced to wait until this management server timed out before being able to connect to the available one.

We fix this by setting the MGM API connection timeout to 5000 milliseconds, which is equal to the timeout used by the TR for getting and setting dynamic ports. (Bug #35714466)

- Values for causes of conflicts used in conflict resolution exceptions tables were misaligned such that the order of `ROW_ALREADY_EXISTS` and `ROW_DOES_NOT_EXIST` was reversed. (Bug #35708719)
- Improved `NDBFS` debugging output for bad requests. (Bug #35500304)

  References: This issue is a regression of: Bug #28922609.
- When other events led to `NDBFS` dumping requests to the log, some of the names of the request types were printed as `Unknown action`. (Bug #35499931)
- `ndb_restore` did not update compare-as-equal primary key values changed during backup. (Bug #35420131)
- In cases where the distributed global checkpoint (GCP) protocol stops making progress, this is detected and optionally handled by the GCP monitor, with handling as determined by the `TimeBetweenEpochsTimeout` and `TimeBetweenGlobalCheckpointsTimeout` data node parameters.

  The LCP protocol is mostly node-local, but depends on the progress of the GCP protocol at the end of a local checkpoint (LCP); this means that, if the GCP protocol stalls, LCPs may also stall in this state. If the LCP watchdog detects that the LCP is stalled in this end state, it should defer to the GCP monitor to handle this situation, since the GCP Monitor is distribution-aware.

  If no GCP monitor limit is set (`TimeBetweenEpochsTimeout` is equal 0), no handling of GCP stalls is performed by the GCP monitor. In this case, the LCP watchdog was still taking action which could eventually lead to cluster failure; this fix corrects this misbehavior so that the LCP watchdog no longer takes any such action. (Bug #29885899)
- Previously, when a timeout was detected during transaction commit and completion, the transaction coordinator (TC) switched to a serial commit-complete execution protocol, which slowed commit-complete processing for large transactions, affecting `GCP_COMMIT` delays and epoch sizes. Instead of switching in such cases, the TC now continues waiting for parallel commit-complete, periodically logging a transaction summary, with states and nodes involved. (Bug #22602898)

  References: See also: Bug #35260944.

Changes in MySQL NDB Cluster 7.6.27 (5.7.43-ndb-7.6.27) (2023-07-18)

MySQL NDB Cluster 7.6.27 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.43 (see Changes in MySQL 5.7.43 (2023-07-18)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change; NDB Cluster APIs:** The `NdbRecord` interface allows equal changes of primary key values; that is, you can update a primary key value to its current value, or to a value which compares as equal according to the collation rules being used, without raising an error. `NdbRecord` does not itself try to prevent the update; instead, the data nodes check whether a primary key is updated to an unequal value and in this case reject the update with Error 897: `Update attempt of primary key via ndbcluster internal api`.

  Previously, when using any other mechanism than `NdbRecord` in an attempt to update a primary key value, the NDB API returned error 4202 `Set value on tuple key attribute is not allowed`, even setting a value identical to the existing one. With this release, the check when performing updates by other means is now passed off to the data nodes, as it is already by `NdbRecord`.

  This change applies to performing primary key updates with `NdbOperation::setValue()`, `NdbInterpretedCode::write_attr()`, and other methods of these two classes which set column values (including `NdbOperation` methods `incValue()`, `subValue()`, `NdbInterpretedCode` methods `add_val()`, `sub_val()`, and so on), as well as the `OperationOptions::OO_SETVALUE` extension to the `NdbOperation` interface. (Bug #35106292)

- NDB 7.6 is now built with support for OpenSSL 3.0. (WL #15614)

Bugs Fixed

- Backups using `NOWAIT` did not start following a restart of the data node. (Bug #35389533)
- When handling the connection (or reconnection) of an API node, it was possible for data nodes to inform the API node that it was permitted to send requests too quickly, which could result in requests not being delivered and subsequently timing out on the API node with errors such as Error 4008 `Receive from Ndb failed` or Error 4012 `Request ndbd time-out, maybe due to high load or communication problems`. (Bug #35387076)
- Made the following improvements in warning output:
  - Now, in addition to local checkpoint (LCP) elapsed time, the maximum time allowed without any progress is also printed.
  - Table IDs and fragment IDs are undefined and thus not relevant when an LCP has reached `WAIT_END_LCP` state, and are no longer printed at that point.
  - When the maximum limit was reached, the same information was shown twice, as both warning and crash information.

  (Bug #35376705)
- When deferred triggers remained pending for an uncommitted transaction, a subsequent transaction could waste resources performing unnecessary checks for deferred triggers; this could lead to an unplanned shutdown of the data node if the latter transaction had no committable operations.

This was because, in some cases, the control state was not reinitialized for management objects used by DBTC.

We fix this by making sure that state initialization is performed for any such object before it is used. (Bug #35256375)

- A pushdown join between queries featuring very large and possibly overlapping IN() and NOT IN() lists caused SQL nodes to exit unexpectedly. One or more of the IN() (or NOT IN()) operators required in excess of 2500 arguments to trigger this issue. (Bug #35185670, Bug #35293781)
- The buffers allocated for a key of size MAX_KEY_SIZE were of insufficient size. (Bug #35155005)
- Some calls made by the ndbcluster handler to push_warning_printf() used severity level ERROR, which caused an assertion in debug builds. This fix changes all such calls to use severity WARNING instead. (Bug #35092279)
- When a connection between a data node and an API or management node was established but communication was available only from the other node to the data node, the data node considered the other node “live”, since it was receiving heartbeats, but the other node did not monitor heartbeats and so reported no problems with the connection. This meant that the data node assumed wrongly that the other node was (fully) connected.

  We solve this issue by having the API or management node side begin to monitor data node liveness even before receiving the first REGCONF signal from it; the other node sends a REGREQ signal every 100 milliseconds, and only if it receives no REGCONF from the data node in response within 60 seconds is the node reported as disconnected. (Bug #35031303)
- The log contained a high volume of messages having the form DICT: index *index number* stats auto-update requested, logged by the DBDICT block each time it received a report from DBTUX requesting an update. These requests often occur in quick succession during writes to the table, with the additional possibility in this case that duplicate requests for updates to the same index were being logged.

  Now we log such messages just before DBDICT actually performs the calculation. This removes duplicate messages and spaces out messages related to different indexes. Additional debug log messages are also introduced by this fix, to improve visibility of the decisions taken and calculations performed. (Bug #34760437)
- Local checkpoints (LCPs) wait for a global checkpoint (GCP) to finish for a fixed time during the end phase, so they were performed sometimes even before all nodes were started.

  In addition, this bound, calculated by the GCP coordinator, was available only on the coordinator itself, and only when the node had been started (start phase 101).

  These two issues are fixed by calculating the bound earlier in start phase 4; GCP participants also calculate the bound whenever a node joins or leaves the cluster. (Bug #32528899)

Changes in MySQL NDB Cluster 7.6.26 (5.7.42-ndb-7.6.26) (2023-04-19)

MySQL NDB Cluster 7.6.26 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.42 (see Changes in MySQL 5.7.42 (2023-04-18)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **MySQL NDB ClusterJ:** Performance has been improved for accessing tables using a single-column partition key when the column is of type `CHAR` or `VARCHAR`. (Bug #35027961)
- Beginning with this release, `ndb_restore` implements the `--timestamp-printouts` option, which causes all error, info, and debug node log messages to be prefixed with timestamps. (Bug #34110068)

Bugs Fixed

- **Microsoft Windows:** Two memory leaks found by code inspection were removed from `NDB` process handles on Windows platforms. (Bug #34872901)
- **Microsoft Windows:** On Windows platforms, the data node angel process did not detect whether a child data node process exited normally. We fix this by keeping an open process handle to the child and using this when probing for the child's exit. (Bug #34853213)
- **NDB Cluster APIs; MySQL NDB ClusterJ:** MySQL ClusterJ uses a scratch buffer for primary key hash calculations which was limited to 10000 bytes, which proved too small in some cases. Now we `malloc()` the buffer if its size is not sufficient.

  This also fixes an issue with the `Ndb` object methods `startTransaction()` and `computeHash()` in the NDB API: Previously, if either of these methods was passed a temporary buffer of insufficient size, the method failed. Now in such cases a temporary buffer is allocated.

  Our thanks to Mikael Ronström for this contribution. (Bug #103814, Bug #32959894)
- **NDB Cluster APIs:** When dropping an event operation (`NdbEventOperation`) in the NDB API, it was sometimes possible for the dropped event operation to remain visible to the application after instructing the data nodes to stop sending events related to this event operation, but before all pending buffered events were consumed and discarded. This could be observed in certain cases when performing an online alter operation, such as `ADD COLUMN` or `RENAME COLUMN`, along with concurrent writes to the affected table.

  Further analysis showed that the dropped events were accessible when iterating through event operations with `Ndb::getGCIEventOperations()`. Now, this method skips dropped events when called iteratively. (Bug #34809944)
- To reduce confusion between the version of the file format and the version of the cluster which produced the backup, the backup file format version is now shown by `ndb_restore` using hexadecimal notation. (Bug #35079426)

  References: This issue is a regression of: Bug #34110068.

- Removed a memory leak in the DBDICT kernel block caused when an internal foreign key definition record was not released when no longer needed. This could be triggered by either of the following events:

  - Drop of a foreign key constraint on an NDB table
  - Rejection of an attempt to create a foreign key constraint on an NDB table

  Such records use the DISK_RECORDS memory resource; you can check this on a running cluster by executing SELECT node_id, used FROM ndbinfo.resources WHERE resource_name='DISK_RECORDS' in the mysql client. This resource uses SharedGlobalMemory, exhaustion of which could lead not only to the rejection of attempts to create foreign keys, but of queries making use of joins as well, since the DBSPJ block also uses shared global memory by way of QUERY_MEMORY. (Bug #35064142)

- When a transaction coordinator is starting fragment scans with many fragments to scan, it may take a realtime break (RTB) during the process to ensure fair CPU access for other requests. When the requesting API disconnected and API failure handling for the scan state occurred before the RTB continuation returned, continuation processing could not proceed because the scan state had been removed.

  We fix this by adding appropriate checks on the scan state as part of the continuation process. (Bug #35037683)

- Sender and receiver signal IDs were printed in trace logs as signed values even though they are actually unsigned 32-bit numbers. This could result in confusion when the top bit was set, as it cuased such numbers to be shown as negatives, counting upwards from -MAX_32_BIT_SIGNED_INT. (Bug #35037396)

- A fiber used by the DICT block monitors all indexes, and triggers index statistics calculations if requested by DBTUX index fragment monitoring; these calculations are performed using a schema transaction. When the DICT fiber attempts but fails to seize a transaction handle for requesting a schema transaction to be started, fiber exited, so that no more automated index statistics updates could be performed without a node failure. (Bug #34992370)

  References: See also: Bug #34007422.

- Schema objects in NDB use composite versioning, comprising major and minor subversions. When a schema object is first created, its major and minor versions are set; when an existing schema object is altered in place, its minor subversion is incremented.

  At restart time each data node checks schema objects as part of recovery; for foreign key objects, the versions of referenced parent and child tables (and indexes, for foreign key references not to or from a table's primary key) are checked for consistency. The table version of this check compares only major subversions, allowing tables to evolve, but the index version also compares minor subversions; this resulted in a failure at restart time when an index had been altered.

  We fix this by comparing only major subversions for indexes in such cases. (Bug #34976028)

  References: See also: Bug #21363253.

- When running an NDB Cluster with multiple management servers, termination of the ndb_mgmd processes required an excessive amount of time when shutting down the cluster. (Bug #34872372)

- When requesting a new global checkpoint (GCP) from the data nodes, such as by the NDB Cluster handler in mysqld to speed up delivery of schema distribution events and responses, the request was

sent 100 times. While the DBDIH block attempted to merge these duplicate requests into one, it was possible on occasion to trigger more than one immediate GCP. (Bug #34836471)

- After receiving a SIGTERM signal, ndb_mgmd did not wait for all threads to shut down before exiting. (Bug #33522783)

  References: See also: Bug #32446105.

- When started with no connection string on the command line, ndb_waiter printed Connecting to mgmsrv at (null). Now in such cases, it prints Connecting to management server at nodeid=0,localhost:1186 if no other default host is specified.

  The --help option and other ndb_waiter program output was also improved. (Bug #12380163)

Changes in MySQL NDB Cluster 7.6.25 (5.7.41-ndb-7.6.25) (2023-01-18)

MySQL NDB Cluster 7.6.25 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.41 (see Changes in MySQL 5.7.41 (2023-01-17)).

Bugs Fixed

- **MySQL NDB ClusterJ:** ClusterJ could not be built on Ubuntu 22.10 with GCC 12.2. (Bug #34666985)

- In some contexts, a data node process may be sent SIGCHLD by other processes. Previously, the data node process bound a signal handler treating this signal as an error, which could cause the process to shut down unexpectedly when run in the foreground in a Kubernetes environment (and possibly under other conditions as well). This occurred despite the fact that a data node process never starts child processes itself, and thus there is no need to take action in such cases.

  To fix this, the handler has been modified to use SIG_IGN, which should result in cleanup of any child processes.

  Note

  mysqld and ndb_mgmd processes do not bind any handlers for SIGCHLD.

  (Bug #34826194)

- The running node from a node group scans each fragment (CopyFrag) and sends the rows to the starting peer in order to synchronize it. If a row from the fragment is locked exclusively by a user transaction, it blocks the scan from reading the fragment, causing the copyFrag to stall.

  If the starting node fails during the CopyFrag phase then normal node failure handling takes place. The cordinator node's transaction coordinator (TC) performs TC takeover of the user transactions from the TCs on the failed node. Since the scan that aids copying the fragment data over to the starting node is

considered internal only, it is not a candidate for takeover, thus the takeover TC marks the `CopyFrag` scan as closed at the next opportunity, and waits until it is closed.

The current issue arose when the `CopyFrag` scan was in the `waiting for row lock` state, and the closing of the marked scan was not performed. This led to TC takeover stalling while waiting for the close, causing unfinished node failure handling, and eventually a GCP stall potentially affecting redo logging, local checkpoints, and NDB Replication.

We fix this by closing the marked `CopyFrag` scan whenever a node failure occurs while the `CopyFrag` is waiting for a row lock. (Bug #34823988)

References: See also: Bug #35037327.

- In certain cases, invalid signal data was not handled correctly. (Bug #34787608)
- Following execution of `DROP NODEGROUP` in the management client, attempting to creating or altering an `NDB` table specifying an explicit number of partitions or using `MAX_ROWS` was rejected with `Got error 771 'Given NODEGROUP doesn't exist in this cluster' from NDB`. (Bug #34649576)
- In a cluster with multiple management nodes, when one management node connected and later disconnected, any remaining management nodes were not aware of this node and were eventually forced to shut down when stopped nodes reconnected; this happened whenever the cluster still had live data nodes.

  On investigation it was found that node disconnection handling was done in the `NF_COMPLETEREP` path in `ConfigManager` but the expected `NF_COMPLETEREP` signal never actually arrived. We solve this by handling disconnecting management nodes when the `NODE_FAILREP` signal arrives, rather than waiting for `NF_COMPLETEREP`. (Bug #34582919)
- When reorganizing a table with `ALTER TABLE ... REORGANIZE PARTITION` following addition of new data nodes to the cluster, unique hash indexes were not redistributed properly. (Bug #30049013)
- During a rolling restart of a cluster with two data nodes, one of them refused to start, reporting that the redo log fragment file size did not match the configured one and that an initial start of the node was required. Fixed by addressing a previously unhandled error returned by `fsync()`, and retrying the write. (Bug #28674694)
- For a partial local checkpoint, each fragment LCP must be to be able to determine the precise state of the fragment at the start of the LCP and the precise difference in the fragment between the start of the current LCP and the start of the previous one. This is tracked using row header information and page header information; in cases where physical pages are removed this is also tracked in logical page map information.

  A page included in the current LCP, before the LCP scan reaches it, is released due to the commit or rollback of some operation on the fragment, also releasing the last used storage on the page.

  Since the released page could not be found by the scan, the release itself set the `LCP_SCANNED_BIT` of the page map entry it was mapped into, in order to indicate that the page was already handled from the point of view of the current LCP, causing subsequent allocation and release of the pages mapped to the entry during the LCP to be ignored. The state of the entry at the start of the LCP was also set as allocated in the page map entry.

  These settings are cleared only when the next LCP is prepared. Any page release associated with the page map entry before the clearance would violate the requirement that the bit is not set; we resolve this issue by removing the (incorrect) requirement. (Bug #23539857)

- A data node could hit an overly strict assertion when the thread liveness watchdog triggered while the node was already shutting down. We fix the issue by relaxing this assertion in such cases. (Bug #22159697)
- Removed a leak of long message buffer memory that occurred each time an index was scanned for updating index statistics. (Bug #108043, Bug #34568135)
- Fixed an uninitialized variable in `Suma.cpp`. (Bug #106081, Bug #33764143)

Changes in MySQL NDB Cluster 7.6.24 (5.7.40-ndb-7.6.24) (2202-10-12)

MySQL NDB Cluster 7.6.24 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.40 (see Changes in MySQL 5.7.40 (2022-10-11)).

Bugs Fixed

- `NdbScanOperation` errors are returned asynchronously to the client, possibly while the client is engaged in other processing. A successful call to `NdbTransaction::execute()` guarantees only that the scan request has been assembled and sent to the transaction coordinator without any errors; it does not wait for any sort of `CONF` or `REF` signal to be returned from the data nodes. In this particular case, the expected `TAB_SCANREF` signal was returned asynchronously into the client space, possibly while the client was still performing other operations.

  We make this behavior more deterministic by not setting the `NdbTransaction` error code when a `TAB_SCANREF` error is received. (Bug #34348706)

- The combination of batching with multiple in-flight operations per key, use of `IgnoreError`, and transient errors occurring on non-primary replicas led in some cases to inconsistencies within `DBTUP` resulting in replica misalignment and other issues. We now prevent this from happening by detecting when operations are failing on non-primary replicas, and forcing `AbortOnError` handling (rollback) in such cases for the containing transaction. (Bug #34013385)

- When the rate of changes was high, event subscribers were slow to acknowledge receipt, or both, it was possible for the `SUMA` block to run out of space for buffering events. (Bug #30467140)

- `ALTER TABLE ... COMMENT="NDB_TABLE=READ_BACKUP=1"` or `ALTER TABLE..COMMENT="NDB_TABLE=READ_BACKUP=0"` performs a non-copying (online) `ALTER` operation on a table to add or remove its `READ_BACKUP` property (see NDB_TABLE Options), which increments the index version of all indexes on the table. Existing statistics, stored using the previous index version, were orphaned and never deleted; this led to wasted memory and inefficient searches when collecting index statistics.

  We address these issues by cleaning up the index samples; we delete any samples whose sample version is greater than or less than the current sample version. In addition, when no existing statistics

are found by index ID and version, and when indexes are dropped. In this last case, we relax the bounds for the delete operation and remove all entries corresponding to the index ID in question, as opposed to both index ID and index version.

This fix cleans up the sample table which stores the bulk of index statistics data. The head table, which consists of index metadata rather than actual statistics, still contains orphaned rows, but since these occupy an insignificant amount of memory, they do not adversely affect statistics search efficiency, and stale entries are cleaned up when index IDs and versions are reused.

See also NDB API Statistics Counters and Variables. (Bug #29611297)

Changes in MySQL NDB Cluster 7.6.23 (5.7.39-ndb-7.6.23) (2022-07-27)

MySQL NDB Cluster 7.6.23 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.39 (see Changes in MySQL 5.7.39 (2022-07-26)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change; NDB Replication:** The `slave_allow_batching` system variable affects how efficiently the slave applies epoch transactions. When this variable is set to `OFF`, by default, every discrete replication event in the binary log is applied and executed separately, which generally leads to poor performance.

  Beginning with this release, if slave batching is disabled (`slave_allow_batching` set to `OFF`), the MySQL server now reports a warning, with an admonition to enable it.

Bugs Fixed

- **NDB Replication:** Updating a row of an `NDB` table having only the hidden primary key (but no explicit PK) on the source could lead to a stoppage of the SQL thread on the replica. (Bug #33974581)
- **NDB Cluster APIs:** The internal function `NdbThread_SetThreadPrio()` sets the thread priority (`thread_prio`) for a given thread type when applying the setting of the `ThreadConfig` configuration parameter. It was possible for this function in some cases to return an error when it had actually succeeded, which could have a an unfavorable impact on the performance of some NDB API applications. (Bug #34038630)
- Path lengths were not always calculated correctly by the data nodes. (Bug #33993607)
- The internal function `NdbReceiver::unpackRecAttr()`, which unpacks attribute values from a buffer from a `GSN_TRANSID_AI` signal, did not check to ensure that attribute sizes fit within the buffer.

This could corrupt the buffer which could in turn lead to reading beyond the buffer and copying beyond destination buffers. (Bug #33941167)

- Some NDB internal signals were not always checked properly. (Bug #33896428)

Changes in MySQL NDB Cluster 7.6.22 (5.7.38-ndb-7.6.22) (2022-04-27)

MySQL NDB Cluster 7.6.22 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.38 (see Changes in MySQL 5.7.38 (2022-04-26)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- The client receive thread was enabled only when under high load, where the criterion for determining “high load” was that the number of clients waiting in the poll queue (the receive queue) was greater than `min_active_clients_recv_thread` (default: `8`).

  This was a poor metric for determining high load, since a single client, such as the binary log injector thread handling incoming replication events, could experience high load on its own as well. The same was true of a pushed join query (in which very large batches of incoming `TRANSID_AI` signals are received).

  We change the receive thread such that it now sleeps in the poll queue rather than being deactivated completely, so that it is now always available for handling incoming signals, even when the client is not under high load. (Bug #33752914)

Bugs Fixed

- **Important Change:** The maximum value supported for the `--ndb-batch-size` server option has been increased from `31536000` to `2147483648` (2 GB). (Bug #21040523)

- **Performance:** When inserting a great many rows into an empty or small table in the same transaction, the rate at which rows were inserted quickly declined to less than 50% of the initial rate; subsequently, it was found that roughly 50% of all CPU time was spent in `Dbacc::getElement()`, and the root cause identified to be the timing of resizing the structures used for storing elements by `DBACC`, growing with the insertion of more rows in the same transaction, and shrinking following a commit.

  We fix this issue by checking for a need to resize immediately following the insertion or deletion of an element. This also handles the subsequent rejection of an insert. (Bug #33803487)

  References: See also: Bug #33803541.

- **Performance:** The internal function `computeXorChecksum()` was implemented such that great care was taken to aid the compiler in generating optimal code, but it was found that it consumed excessive CPU resources, and did not perform as well as a simpler implementation. This function is now reimplemented with a loop summing up `XOR` results over an array, which appears to result in better optimization with both GCC and Clang compilers. (Bug #33757412)

- 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)

- `NdbEventBuffer` hash key generation for non-character data reused the same 256 hash keys; in addition, strings of zero length were ignored when calculating hash keys. (Bug #33783274)

- The collection of NDB API statistics based on the `EventBytesRecvdCount` event counter incurred excessive overhead. Now this counter is updated using a value which is aggregated as the event buffer is filled, rather than traversing all of the event buffer data in a separate function call.

  For more information, see NDB API Statistics Counters and Variables. (Bug #33778923)

- The receiver thread ID was hard-coded in the internal method `TransporterFacade::raise_thread_prio()` such that it always acted to raise the priority of the receiver thread, even when called from the send thread. (Bug #33752983)

- The deprecated `-r` option for `ndbd` has been removed. In addition, this change also removes extraneous text from the output of `ndbd --help`. (Bug #33362935)

  References: See also: Bug #31565810.

- The `ndb_import` tool handled only the hidden primary key which is defined by `NDB` when a table does not have an explicit primary key. This caused an error when inserting a row containing `NULL` for an auto-increment primary key column, even though the same row was accepted by `LOAD DATA INFILE`.

  We fix this by adding support for importing a table with one or more instances of `NULL` in an auto-increment primary key column. This includes a check that a table has no more than one auto-increment column; if this column is nullable, it is redefined by `ndb_import` as `NOT NULL`, and any occurrence of `NULL` in this column is replaced by a generated auto-increment value before inserting the row into `NDB`. (Bug #30799495)

- When a node failure is detected, surviving nodes in the same nodegroup as this node attempt to resend any buffered change data to event subscribers. In cases in which there were no outstanding epoch deliveries, that is, the list of unacknowledged GCIs was empty, the surviving nodes made the incorrect assumption that this list would never be empty. (Bug #30509416)

Changes in MySQL NDB Cluster 7.6.21 (5.7.37-ndb-7.6.21) (2022-01-19)

MySQL NDB Cluster 7.6.21 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.37 (see Changes in MySQL 5.7.37 (2022-01-18)).

Bugs Fixed

- **Important Change:** The deprecated data node option `--connect-delay` has been removed. This option was a synonym for `--connect-retry-delay`, which was not honored in all cases; this issue has been fixed, and the option now works correctly. In addition, the short form `-r` for this option has been deprecated, and you should expect it to be removed in a future release. (Bug #31565810)

  References: See also: Bug #33362935.

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

- Timing of `wait for scans` log output during online reorganization was not performed correctly. As part of this fix, we change timing to generate one message every 10 seconds rather than scaling indefinitely, so as to supply regular updates. (Bug #35523977)

- Added missing values checks in `ndbd` and `ndbmtd`. (Bug #33661024)

- Online table reorganization increases the number of fragments of a table, and moves rows between them. This is done in the following steps:

  1. Copy rows to new fragments
  2. Update distribution information (hashmap count and total fragments)
  3. Wait for scan activity using old distribution to stop
  4. Delete rows which have moved out of existing partitions
  5. Remove reference to old hashmap
  6. Wait for scan activity started since step 2 to stop

  Due to a counting error, it was possible for the reorganization to hang in step 6; the scan reference count was not decremented, and thus never reached zero as expected. (Bug #33523991)

- The same pushed join on `NDB` tables returned an incorrect result when the `batched_key_access` optimizer switch was enabled.

  This issue arose as follows: When the batch key access (BKA) algorithm is used to join two tables, a set of batched keys is first collected from one of the tables; a multirange read (MRR) operation is constructed against the other. A set of bounds (ranges) is specified on the MRR, using the batched keys to construct each bound.

  When result rows are returned it is necessary to identify which range each returned row comes from. This is used to identify the outer table row to perform the BKA join with. When the MRR operation in question was a root of a pushed join operation, `SPJ` was unable to retrieve this identifier (`RANGE_NO`). We fix this by implementing the missing `SPJ` API functionality for returning such a `RANGE_NO` from a pushed join query. (Bug #33416308)

- The MySQL Optimizer uses two different methods, `handler::read_cost()` and `Cost_model::page_read_cost()`, to estimate the cost for different access methods, but the cost values returned by these were not always comparable; in some cases this led to the wrong index being

chosen and longer execution time for effected queries. To fix this for NDB, we override the optimizer's page_read_cost() method with one specific to NDBCLUSTER. It was also found while working on this issue that the NDB handler did not implement the read_time() method, used by read_cost(); this method is now implemented by ha_ndbcluster, and thus the optimizer can now properly take into account the cost difference for NDB when using a unique key as opposed to an ordered index (range scan). (Bug #33317872)

- In certain cases, an event's category was not properly detected. (Bug #33304814)
- DBDICT did not always perform table name checks correctly. (Bug #33161548)
- Added a number of missing ID and other values checks in ndbd and ndbmtd. (Bug #33161486, Bug #33162047)
- Added a number of missing ID and other values checks in ndbd and ndbmtd. (Bug #33161259, Bug #33161362)
- 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)
- The order of parameters used in the argument to ndb_import --csvopt is now handled consistently, with the rightmost parameter always taking precedence. This also applies to duplicate instances of a parameter. (Bug #32822757)
- In some cases, issues with the redo log while restoring a backup led to an unplanned shutdown of the data node. To fix this, when the redo log file is not available for writes, we now include the correct wait code and waiting log part in the CONTINUEB signal before sending it. (Bug #32733659)

  References: See also: Bug #31585833.
- When a local data manager had no fragments, the data node could not be restarted. While it is unusual, this lack of any fragments can occur when there have been changes in the number of LDMs since any tables were last created. (Bug #32288569)
- A query used by MySQL Enterprise Monitor to monitor memory use in NDB Cluster became markedly less performant as the number of NDB tables increased. We fix this as follows:
  - Row counts for virtual ndbinfo tables have been made available to the MySQL optimizer
  - Size estimates are now provided for all ndbinfo tables

  Following these improvements, queries against ndbinfo tables should be noticeably faster. (Bug #28658625)
- NDB did not close any pending schema transactions when returning an error from internal system table creation and drop functions.

Changes in MySQL NDB Cluster 7.6.20 (5.7.36-ndb-7.6.20) (2021-10-20)

MySQL NDB Cluster 7.6.20 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.36 (see Changes in MySQL 5.7.36 (2021-10-19)).

Bugs Fixed

- 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)
- Added an ndbrequire() in QMGR to ensure that process reports from signal data use appropriate node IDs. (Bug #32983240)
- 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)

- Event buffer status messages shown by the event logger have been improved. Percentages are now displayed only when it makes to do so. In addition, if a maximum size is not defined, the printout shows `max=unlimited`. (Bug #21276857)

Changes in MySQL NDB Cluster 7.6.19 (5.7.35-ndb-7.6.19) (2021-07-21)

MySQL NDB Cluster 7.6.19 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.35 (see Changes in MySQL 5.7.35 (2021-07-20)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- `ndb_restore` now supports conversion between `NULL` and `NOT NULL` columns, as follows:

  - To restore a `NULL` column as `NOT NULL`, use the `--lossy-conversions` option. The presence of any `NULL` rows in the column causes `ndb_restore` to raise an and exit.

  - To restore a `NOT NULL` column as `NULL`, use the `--promote-attributes` option.

  For more information, see the descriptions of the indicated `ndb_restore` options. (Bug #32702637)

Bugs Fixed

- **Packaging:** The `ndb-common` man page was removed, and the information it contained moved to other man pages. (Bug #32799519)

- **NDB Cluster APIs:** Added the `NDB_LE_EventBufferStatus3` log event type to `Ndb_logevent_type` in the MGM API. This is an extension of the `NDB_LE_EventBufferStatus` type which handles total, maximum, and allocated bytes as 64-bit values.

  As part of this fix, the maximum value of the `ndb_eventbuffer_max_alloc` server system variable is increased to 9223372036854775807 (2^{63} - 1).

  For more information, see The Ndb_logevent_type Type. (Bug #32381666)

- `Ndb_rep_tab_key` member variables were not null-terminated before being logged. (Bug #32841430)

References: See also: Bug #32393245.

- Some error messages printed by `ndb_restore` tried to access transactions that were already closed for error information, resulting in an unplanned exit. (Bug #32815725)

- Returning an error while determining the number of partitions used by a `NDB` table caused the MySQL server to write `Incorrect information in table.frm file` to its error log, despite the fact that the indicated file did not exist. This also led to problems with flooding of the error log when users attempted to open `NDB` tables while the MySQL server was not actually connected to `NDB`.

  We fix this by changing the function that determines the number of partitions to return 0 whenever `NDB` is not available, thus deferring any error detection until the MySQL server is once again connected to `NDB`. (Bug #32713166)

- Using duplicate node IDs with `CREATE NODEGROUP` (for example, `CREATE NODEGROUP 11, 11`) could lead to an unplanned shutdown of the cluster. Now when this command includes duplicate node IDs, it raises an error. (Bug #32701583)

- Improved the performance of queries against the `ndbinfo.cluster_locks` table, which could in some cases run quite slowly. (Bug #32655988)

- It was possible in some cases to miss the end point of undo logging for a fragment. (Bug #32623528)

  References: See also: Bug #31774459.

- The `--resume` option for `ndb_import` did not work correctly unless the `--stats` option was also specified. (Bug #31107058)

- A `DELETE` statement whose `WHERE` clause referred to a `BLOB` column was not executed correctly. (Bug #13881465)

Changes in MySQL NDB Cluster 7.6.18 (5.7.34-ndb-7.6.18) (2021-04-21)

MySQL NDB Cluster 7.6.18 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.34 (see Changes in MySQL 5.7.34 (2021-04-20)).

Bugs Fixed

- `ndb_mgmd` now exits gracefully in the event of a `SIGTERM` just as it does following a management client `SHUTDOWN` command. (Bug #32446105)

- A node was permitted during a restart to participate in a backup before it had completed recovery, instead of being made to wait until its recovery was finished. (Bug #32381165)

- Running out of disk space while performing an `NDB` backup could lead to an unplanned shutdown of the cluster. (Bug #32367250)
- The default number of partitions per node (shown in `ndb_desc` output as `PartitionCount`) is calculated using the lowest number of LDM threads employed by any single live node, and was done only once, even after data nodes left or joined the cluster, possibly with a new configuration changing the LDM thread count and thus the default partition count. Now in such cases, we make sure the default number of partitions per node is recalculated whenever data nodes join or leave the cluster. (Bug #32183985)
- The local checkpoint (LCP) mechanism was changed in NDB 7.6 such that it also detected idle fragments—that is, fragments which had not changed since the last LCP and thus required no on-disk metadata update. The LCP mechanism could then immediately proceed to handle the next fragment. When there were a great many such idle fragments, the CPU consumption required merely to loop through these became highly significant, causing latency spikes in user transactions.

  A 1 ms delay was already inserted between each such idle fragment being handled. Testing later showed this to be too short an interval, and that we are normally not in as great a hurry to complete these idle fragments as we previously believed.

  This fix extends the idle fragment delay time to 20 ms if there are no redo alerts indicating an urgent need to complete the LCP. In case of a low redo alert state we wait 5 ms instead, and for a higher alert state we fall back to the 1 ms delay. (Bug #32068551)

  References: See also: Bug #31655158, Bug #31613158.
- Event buffer congestion could lead to unplanned shutdown of SQL nodes which were performing binary logging. We fix this by updating the binary logging handler to use `Ndb::pollEvents2()` (rather than the deprecated `pollEvents()` method) to catch and handle such errors properly, instead. (Bug #31926584)
- Generation of internal statistics relating to `NDB` object counts was found to lead to an increase in transaction latency at very high rates of transactions per second, brought about by returning an excessive number of freed `NDB` objects. (Bug #31790329)
- Calculation of the redo alert state based on redo log usage was overly aggressive, and thus incorrect, when using more than 1 log part per LDM.

Changes in MySQL NDB Cluster 7.6.17 (5.7.33-ndb-7.6.17) (2021-01-19)

MySQL NDB Cluster 7.6.17 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.33 (see Changes in MySQL 5.7.33 (2021-01-18)).

- 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

- **NDB Replication:** After issuing `RESET SLAVE ALL`, `NDB` failed to detect that the replica had restarted. (Bug #31515760)

- While retrieving sorted results from a pushed-down join using `ORDER BY` with the `index` access method (and without `filesort`), an SQL node sometimes unexpectedly terminated. (Bug #32203548)

- Logging of redo log initialization showed log part indexes rather than log part numbers. (Bug #32200635)

- Signal data was overwritten (and lost) due to use of extended signal memory as temporary storage. Now in such cases, extended signal memory is not used in this fashion. (Bug #32195561)

- 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.

- As with writing redo log records, when the file currently used for writing global checkpoint records becomes full, writing switches to the next file. This switch is not supposed to occur until the new file is actually ready to receive the records, but no check was made to ensure that this was the case. This could lead to an unplanned data node shutdown restoring data from a backup using `ndb_restore`. (Bug #31585833)

- `ndb_restore` encountered intermittent errors while replaying backup logs which deleted blob values; this was due to deletion of blob parts when a main table row containing blob one or more values was deleted. This is fixed by modifying `ndb_restore` to use the asynchronous API for blob deletes, which does not trigger blob part deletes when a blob main table row is deleted (unlike the synchronous API), so that a delete log event for the main table deletes only the row from the main table. (Bug #31546136)

- 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 7.6.16 (5.7.32-ndb-7.6.16) (2020-10-20)

MySQL NDB Cluster 7.6.16 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.32 (see Changes in MySQL 5.7.32 (2020-10-19)).

- 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)

- **NDB Cluster APIs:** In certain cases, the Table::getColumn() method returned the wrong Column object. This could happen when the full name of one table column was a prefix of the name of another, or when the names of two columns had the same hash value. (Bug #31774685)

- **NDB Cluster APIs:** It was possible to make invalid sequences of NDB API method calls using blobs. This was because some method calls implicitly cause transaction execution inline, to deal with blob parts and other issues, which could cause user-defined operations not to be handled correctly due to the use of a method executing operations relating to blobs while there still user-defined blob operations pending. Now in such cases, NDB raises a new error 4558 Pending blob operations must be executed before this call. (Bug #27772916)

- After encountering the data node in the configuration file which used `NodeGroup=65536`, the management server stopped assigning data nodes lacking an explicit `NodeGroup` setting to node groups. (Bug #31825181)
- In some cases, `QMGR` returned conflicting `NDB` engine and MySQL server version information, which could lead to unplanned management node shutdown. (Bug #31471959)
- During different phases of the restore process, `ndb_restore` used different numbers of retries for temporary errors as well as different sleep times between retries. This is fixed by implementing consistent retry counts and sleep times across all restore phases. (Bug #31372923)
- Backups errored out with **`FsErrInvalidParameters`** when the filesystem was running with `O_DIRECT` and a data file write was not aligned with the 512-byte block size used by `O_DIRECT` writes. If the total fragment size in the data file is not aligned with the `O_DIRECT` block size, `NDB` pads the last write to the required size, but when there were no fragments to write, `BACKUP` wrote only the header and footer to the data file. Since the header and footer are less than 512 bytes, leading to the issue with the `O_DIRECT` write.

  This is fixed by padding out the generic footer to 512 bytes if necessary, using an `EMPTY_ENTRY`, when closing the data file. (Bug #31180508)
- Altering the table comment of a fully replicated table using `ALGORITHM=INPLACE` led to an assertion. (Bug #31139313)
- Data nodes did not start when the `RealtimeScheduler` configuration parameter was set to 1. This was due to the fact that index builds during startup are performed by temporarily diverting some I/O threads for use as index building threads, and these threads inherited the realtime properties of the I/O threads. This caused a conflict (treated as a fatal error) when index build thread specifications were checked to ensure that they were not realtime threads. This is fixed by making sure that index build threads are not treated as realtime threads regardless of any settings applying to their host I/O threads, which is as actually intended in their design. (Bug #27533538)

Changes in MySQL NDB Cluster 7.6.15 (5.7.31-ndb-7.6.15) (2020-07-14)

MySQL NDB Cluster 7.6.15 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.31 (see Changes in MySQL 5.7.31 (2020-07-13)).

Bugs Fixed

- **NDB Disk Data:** `ndbmtd` sometimes terminated unexpectedly when it could not complete a lookup for a log file group during a restore operation. (Bug #31284086)
- **NDB Disk Data:** An uninitialized variable led to issues when performing Disk Data DDL operations following a restart of the cluster. (Bug #30592528)

- `ndb_restore --remap-column` did not handle columns containing `NULL` values correctly. Now any offset specified by the mapping function used with this option is not applied to `NULL`, so that `NULL` is preserved as expected. (Bug #31966676)

- `MaxDiskWriteSpeedOwnRestart` was not honored as an upper bound for local checkpoint writes during a node restart. (Bug #31337487)

  References: See also: Bug #29943227.

- 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)

- `DUMP 1001` (`DumpPageMemoryOnFail`) now prints out information about the internal state of the data node page memory manager when allocation of pages fails due to resource constraints. (Bug #31231286)

- Statistics generated by `NDB` for use in tracking internal objects allocated and deciding when to release them were not calculated correctly, with the result that the threshold for resource usage was 50% higher than intended. This fix corrects the issue, and should allow for reduced memory usage. (Bug #31127237)

- 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)

- When responding to a `SCANTABREQ`, an API node can provide a distribution key if it knows that the scan should work on only one fragment, in which case the distribution key should be the fragment ID, but in some cases a hash of the partition key was used instead, leading to failures in `DBTC`. (Bug #30774226)

- Several memory leaks found in `ndb_import` have been removed. (Bug #30756434, Bug #30727956)

- The master node in a backup shut down unexpectedly on receiving duplicate replies to a `DEFINE_BACKUP_REQ` signal. These occurred when a data node other than the master errored out during the backup, and the backup master handled the situation by sending itself a `DEFINE_BACKUP_REF` signal on behalf of the missing node, which resulted in two replies being received from the same node (a `CONF` signal from the problem node prior to shutting down and the `REF` signal from the master on behalf of this node), even though the master expected only one reply per node. This scenario was also encountered for `START_BACKUP_REQ` and `STOP_BACKUP_REQ` signals.

  This is fixed in such cases by allowing duplicate replies when the error is the result of an unplanned node shutdown. (Bug #30589827)

- When updating `NDB_TABLE` comment options using `ALTER TABLE`, other options which has been set to non-default values when the table was created but which were not specified in the `ALTER TABLE` statement could be reset to their defaults.

  See Setting NDB Comment Options, for more information. (Bug #30428829)

- When, during a restart, a data node received a `GCP_SAVEREQ` signal prior to beginning start phase 9, and thus needed to perform a global checkpoint index write to a local data manager's local checkpoint control file, it did not record information from the `DIH` block originating with the node that sent the signal as part of the data written. This meant that, later in start phase 9, when attempting to send a `GCP_SAVECONF` signal in response to the `GCP_SAVEREQ`, this information was not available, which meant the response could not be sent, resulting in an unplanned shutdown of the data node. (Bug #30187949)

- Setting `EnableRedoControl` to `false` did not fully disable `MaxDiskWriteSpeed`, `MaxDiskWriteSpeedOtherNodeRestart`, and `MaxDiskWriteSpeedOwnRestart` as expected. (Bug #29943227)

  References: See also: Bug #31337487.

- Removed a memory leak found in the `ndb_import` utility. (Bug #29820879)

- A `BLOB` value is stored by `NDB` in multiple parts; when reading such a value, one read operation is executed per part. If a part is not found, the read fails with a `row not found error`, which indicates a corrupted `BLOB`, since a `BLOB` should never have any missing parts. A problem can arise because this error is reported as the overall result of the read operation, which means that mysqld sees no error and reports zero rows returned.

  This issue is fixed by adding a check specifically for the case in wich a blob part is not found. Now, when this occurs, overwriting the `row not found` error with `corrupted blob`, which causes the originating `SELECT` statement to fail as expected. Users of the NDB API should be aware that, despite this change, the `NdbBlob::getValue()` method continues to report the error as `row not found` in such cases. (Bug #28590428)

- Incorrect handling of operations on fragment replicas during node restarts could result in a forced shutdown, or in content diverging between fragment replicas, when primary keys with nonbinary (case-sensitive) equality conditions were used. (Bug #98526, Bug #30884622)

Changes in MySQL NDB Cluster 7.6.14 (5.7.30-ndb-7.6.14) (2020-04-28)

MySQL NDB Cluster 7.6.14 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.30 (see Changes in MySQL 5.7.30 (2020-04-27)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **NDB Client Programs:** Two options are added for the `ndb_blob_tool` utility, to enable it to detect missing blob parts for which inline parts exist, and to replace these with placeholder blob parts (consisting of space characters) of the correct length. To check whether there are missing blob parts, use the `ndb_blob_tool --check-missing` option. To replace with placeholders any blob parts which are missing, use the program's `--add-missing` option, also added in this release. (Bug #28583971)

- **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. (WL #13727, WL #13728)

- **MySQL NDB ClusterJ:** The unused `antlr3` plugin has been removed from the ClusterJ `pom` file. (Bug #29931625)

- **MySQL NDB ClusterJ:** The minimum Java version ClusterJ supports for MySQL NDB Cluster 8.0 is now Java 8. (Bug #29931625)

- **MySQL NDB ClusterJ:** A few Java APIs used by ClusterJ are now deprecated in recent Java versions. These adjustments have been made to ClusterJ:

  - Replaced all `Class.newInstance()` calls with `Class.getDeclaredConstructor().newInstance()` calls. Also updated the exception handling and the test cases wherever required.

  - All the `Number` classes' constructors that instantiate an object from a `String` or a primitive type are deprecated. Replaced all such deprecated instantiation calls with the corresponding `valueOf()` method calls.

  - The `Proxy.getProxyClass()` is now deprecated. The `DomainTypeHandlerImpl` class now directly creates a new instance using the `Proxy.newProxyInstance()` method; all references to the `Proxy` class and its constructors are removed from the `DomainTypeHandlerImpl` class. `SessionFactoryImpl` class now uses the interfaces underlying the proxy object to identify the domain class rather than using the Proxy class. Also updated `DomainTypeHandlerFactoryTest`.

  - The `finalize()` method is now deprecated. This patch does not change the overriding finalize() methods, but just suppresses the warnings on them. This deprecation will be handled separately in a later patch.

  - Updated the CMake configuration to treat deprecation warnings as errors when compiling ClusterJ.

(Bug #29931625)

- It now possible to consolidate data from separate instances of NDB Cluster into a single target NDB Cluster when the original datasets all use the same schema. This is supported when using backups created using `START BACKUP` in `ndb_mgm` and restoring them with `ndb_restore`, using the `--remap-column` option implemented in this release (along with `--restore-data` and possibly other options). `--remap-column` can be employed to handle cases of overlapping primary, unique, or both sorts of key values between source clusters, and you need to make sure that they do not overlap in the target cluster. This can also be done to preserve other relationships between tables.

  When used together with `--restore-data`, the new option applies a function to the value of the indicated column. The value set for this option is a string of the format *`db.tbl.col:fn:args`*, whose components are listed here:

  - *`db`*: Database name, after performing any renames.
  - *`tbl`*: Table name.
  - *`col`*: Name of the column to be updated. This column's type must be one of `INT` or `BIGINT`, and can optionally be `UNSIGNED`.
  - *`fn`*: Function name; currently, the only supported name is `offset`.
  - *`args`*: The size of the offset to be added to the column value by `offset`. The range of the argument is that of the signed variant of the column's type; thus, negative offsets are supported.

  You can use `--remap-column` for updating multiple columns of the same table and different columns of different tables, as well as combinations of multiple tables and columns. Different offset values can be employed for different columns of the same table.

  As part of this work, two new options are also added to `ndb_desc` in this release:

  - `--auto-inc` (short form `-a`): Includes the next auto-increment value in the output, if the table has an `AUTO_INCREMENT` column.
  - `--context` (short form `-x`): Provides extra information about the table, including the schema, database name, table name, and internal ID.

  These options may be useful for obtaining information about `NDB` tables when planning a merge, particularly in situations where the `mysql` client may not be readily available.

  For more information, see the descriptions for `--remap-column`, `--auto-inc`, and `--context`. (Bug #30383950, WL #11796)

- `ndb_restore` now supports different primary key definitions for source and target tables when restoring from an `NDB` native backup, using the `--allow-pk-changes` option introduced in this release. Both increasing and decreasing the number of columns making up the original primary key are supported. This may be useful when it is necessary to accommodate schema version changes while restoring data, or when doing so is more efficient or less time-consuming than performing `ALTER TABLE` statements involving primary key changes on a great many tables following the restore operation.

  When extending a primary key with additional columns, any columns added must not be nullable, and any values stored in any such columns must not change while the backup is being taken. Changes in the values of any such column while trying to add it to the table's primary key causes the restore operation to fail. Due to the fact that some applications set the values of all columns when updating a row even if the values of one or more of the columns does not change, it is possible to override this behavior by using

the `--ignore-extended-pk-updates` option which is also added in this release. If you do this, care must be taken to insure that such column values do not actually change.

When removing columns from the table's primary key, it is not necessary that the columns dropped from the primary key remain part of the table afterwards.

For more information, see the description of the `--allow-pk-changes` option in the documentation for `ndb_restore`. (Bug #26435136, Bug #30383947, Bug #30634010, WL #10730)

- 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

- **MySQL NDB ClusterJ:** When a `Date` value was read from a NDB cluster, ClusterJ sometimes extracted the wrong year value from the row. It was because the `Utility` class, when unpacking the `Date` value, wrongly extracted some extra bits for the year. This patch makes ClusterJ only extract the required bits. (Bug #30600320)
- **MySQL NDB ClusterJ:** When the cluster's `NdbOperation::AbortOption` type had the value of `AO_IgnoreOnError`, when there was a read error, ClusterJ took that as the row was missing and returned `null` instead of an exception. This was because with `AO_IgnoreOnErro`, the `execute()` method always returns a success code after each transaction, and ClusterJ is supposed to check for any errors in any of the individual operations; however, read operations were not checked by ClusterJ in the case. With this patch, read operations are now checked for errors after query executions, so that a reading error is reported as such. (Bug #30076276)
- When restoring signed auto-increment columns, `ndb_restore` incorrectly handled negative values when determining the maximum value included in the data. (Bug #30928710)
- When processing a CSV file, `ndb_import` did not accept trailing field terminators at the ends of lines that were accepted by `mysqlimport`. (Bug #30434663)
- 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 7.6.13 (5.7.29-ndb-7.6.13) (2020-01-14)

MySQL NDB Cluster 7.6.13 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.29 (see Changes in MySQL 5.7.29 (2020-01-13)) .

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change:** It is now possible to divide a backup into slices and to restore these in parallel using two new options implemented for the `ndb_restore` utility, making it possible to employ multiple instances of `ndb_restore` to restore subsets of roughly the same size of the backup in parallel, which should help to reduce the length of time required to restore an NDB Cluster from backup.

  The `--num-slices` options determines the number of slices into which the backup should be divided; `--slice-id` provides the ID of the slice (0 to 1 less than the number of slices) to be restored by `ndb_restore`.

  Up to 1024 slices are supported.

  For more information, see the descriptions of the `--num-slices` and `--slice-id` options. (Bug #30383937, WL #10691)

Bugs Fixed

- **Incompatible Change:** The minimum value for the `RedoOverCommitCounter` data node configuration parameter has been increased from 0 to 1. The minimum value for the `RedoOverCommitLimit` data node configuration parameter has also been increased from 0 to 1.

  You should check the cluster global configuration file and make any necessary adjustments to values set for these parameters before upgrading. (Bug #29752703)

- **Microsoft Windows; NDB Disk Data:** On Windows, restarting a data node other than the master when using Disk Data tables led to a failure in `TSMAN`. (Bug #97436, Bug #30484272)

- **NDB Disk Data:** Compatibility code for the Version 1 disk format used prior to the introduction of the Version 2 format in NDB 7.6 turned out not to be necessary, and is no longer used.

- A faulty `ndbrequire()` introduced when implementing partial local checkpoints assumed that `m_participatingLQH` must be clear when receiving `START_LCP_REQ`, which is not necessarily true when a failure happens for the master after sending `START_LCP_REQ` and before handling any `START_LCP_CONF` signals. (Bug #30523457)

- A local checkpoint sometimes hung when the master node failed while sending an `LCP_COMPLETE_REP` signal and it was sent to some nodes, but not all of them. (Bug #30520818)

- Added the `DUMP 9988` and `DUMP 9989` commands. (Bug #30520103)

- Execution of `ndb_restore --rebuild-indexes` together with the `--rewrite-database` and `--exclude-missing-tables` options did not create indexes for any tables in the target database. (Bug #30411122)

- When synchronizing extent pages it was possible for the current local checkpoint (LCP) to stall indefinitely if a `CONTINUEB` signal for handling the LCP was still outstanding when receiving the `FSWRITECONF` signal for the last page written in the extent synchronization page. The LCP could also be restarted if another page was written from the data pages. It was also possible that this issue caused `PREP_LCP` pages to be written at times when they should not have been. (Bug #30397083)

- 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.

- Data node failures with the error `Another node failed during system restart...` occurred during a partial restart. (Bug #30368622)

- If a `SYNC_EXTENT_PAGES_REQ` signal was received by `PGMAN` while dropping a log file group as part of a partial local checkpoint, and thus dropping the page locked by this block for processing next, the LCP terminated due to trying to access the page after it had already been dropped. (Bug #30305315)

- The wrong number of bytes was reported in the cluster log for a completed local checkpoint. (Bug #30274618)

  References: See also: Bug #29942998.

- The number of data bytes for the summary event written in the cluster log when a backup completed was truncated to 32 bits, so that there was a significant mismatch between the number of log records and the number of data records printed in the log for this event. (Bug #29942998)

- Using 2 LDM threads on a 2-node cluster with 10 threads per node could result in a partition imbalance, such that one of the LDM threads on each node was the primary for zero fragments. Trying to restore a multi-threaded backup from this cluster failed because the datafile for one LDM contained only the 12-byte data file header, which `ndb_restore` was unable to read. The same problem could occur in other cases, such as when taking a backup immediately after adding an empty node online.

  It was found that this occurred when `ODirect` was enabled for an EOF backup data file write whose size was less than 512 bytes and the backup was in the `STOPPING` state. This normally occurs only for an aborted backup, but could also happen for a successful backup for which an LDM had no fragments. We fix the issue by introducing an additional check to ensure that writes are skipped only if the backup actually contains an error which should cause it to abort. (Bug #29892660)

  References: See also: Bug #30371389.

- In some cases the `SignalSender` class, used as part of the implementation of `ndb_mgmd` and `ndbinfo`, buffered excessive numbers of unneeded `SUB_GCP_COMPLETE_REP` and `API_REGCONF` signals, leading to unnecessary consumption of memory. (Bug #29520353)

  References: See also: Bug #20075747, Bug #29474136.

- The setting for the `BackupLogBufferSize` configuration parameter was not honored. (Bug #29415012)

- 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.

- A transaction which inserts a child row may run concurrently with a transaction which deletes the parent row for that child. One of the transactions should be aborted in this case, lest an orphaned child row result.

  Before committing an insert on a child row, a read of the parent row is triggered to confirm that the parent exists. Similarly, before committing a delete on a parent row, a read or scan is performed to

confirm that no child rows exist. When insert and delete transactions were run concurrently, their prepare and commit operations could interact in such a way that both transactions committed. This occurred because the triggered reads were performed using `LM_CommittedRead` locks (see `NdbOperation::LockMode`), which are not strong enough to prevent such error scenarios.

This problem is fixed by using the stronger `LM_SimpleRead` lock mode for both triggered reads. The use of `LM_SimpleRead` rather than `LM_CommittedRead` locks ensures that at least one transaction aborts in every possible scenario involving transactions which concurrently insert into child rows and delete from parent rows. (Bug #22180583)

- 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 7.6.12 (5.7.28-ndb-7.6.12) (2019-10-15)

MySQL NDB Cluster 7.6.12 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.28 (see Changes in MySQL 5.7.28 (2019-10-14)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- `ndb_restore` now reports the specific `NDB` error number and message when it is unable to load a table descriptor from a backup `.ctl` file. This can happen when attempting to restore a backup taken from a later version of the NDB Cluster software to a cluster running an earlier version—for example, when the backup includes a table using a character set which is unknown to the version of `ndb_restore` being used to restore it. (Bug #30184265)
- The output from `DUMP 1000` in the `ndb_mgm` client has been extended to provide information regarding total data page usage. (Bug #29841454)

  References: See also: Bug #29929996.

Bugs Fixed

- **NDB Disk Data:** When a data node failed following creation and population of an `NDB` table having columns on disk, but prior to execution of a local checkpoint, it was possible to lose row data from the tablespace. (Bug #29506869)
- **MySQL NDB ClusterJ:** If ClusterJ was deployed as a separate module of a multi-module web application, when the application tried to create a new instance of a domain object, the exception `java.lang.IllegalArgumentException: non-public interface is not defined by the given loader` was thrown. It was because ClusterJ always tries to create a proxy class from which the domain object can be instantiated, and the proxy class is an implementation of the domain

interface and the protected `DomainTypeHandlerImpl::Finalizable` interface. The class loaders of these two interfaces were different in the case, as they belonged to different modules running on the web server, so that when ClusterJ tried to create the proxy class using the domain object interface's class loader, the above-mentioned exception was thrown. This fix makes the `Finalization` interface public so that the class loader of the web application would be able to access it even if it belongs to a different module from that of the domain interface. (Bug #29895213)

- **MySQL NDB ClusterJ:** ClusterJ sometimes failed with a segmentation fault after reconnecting to an NDB Cluster. This was due to ClusterJ reusing old database metadata objects from the old connection. With the fix, those objects are discarded before a reconnection to the cluster. (Bug #29891983)

- Once a data node is started, 95% of its configured `DataMemory` should be available for normal data, with 5% to spare for use in critical situations. During the node startup process, all of its configured `DataMemory` is usable for data, in order to minimize the risk that restoring the node data fails due to running out of data memory due to some dynamic memory structure using more pages for the same data than when the node was stopped. For example, a hash table grows differently during a restart than it did previously, since the order of inserts to the table differs from the historical order.

  The issue raised in this bug report occurred when a check that the data memory used plus the spare data memory did not exceed the value set for `DataMemory` failed at the point where the spare memory was reserved. This happened as the state of the data node transitioned from starting to started, when reserving spare pages. After calculating the number of reserved pages to be used for spare memory, and then the number of shared pages (that is, pages from shared global memory) to be used for this, the number of reserved pages already allocated was not taken into consideration. (Bug #30205182)

  References: See also: Bug #29616383.

- When executing a global schema lock (GSL), `NDB` used a single `Ndb_table_guard` object for successive retires when attempting to obtain a table object reference; it was not possible for this to succeed after failing on the first attempt, since `Ndb_table_guard` assumes that the underlying object pointer is determined once only—at initialisation—with the previously retrieved pointer being returned from a cached reference thereafter.

  This resulted in infinite waits to obtain the GSL, causing the binlog injector thread to hang so that `mysqld` considered all `NDB` tables to be read-only. To avoid this problem, `NDB` now uses a fresh instance of `Ndb_table_guard` for each such retry. (Bug #30120858)

  References: This issue is a regression of: Bug #30086352.

- When starting, a data node's local sysfile was not updated between the first completed local checkpoint and start phase 50. (Bug #30086352)

- In the `BACKUP` block, the assumption was made that the first record in `c_backups` was the local checkpoint record, which is not always the case. Now `NDB` loops through the records in `c_backups` to find the (correct) LCP record instead. (Bug #30080194)

- During node takeover for the master it was possible to end in the state `LCP_STATUS_IDLE` while the remaining data nodes were reporting their state as `LCP_TAB_SAVED`. This led to failure of the node when attempting to handle reception of a `LCP_COMPLETE_REP` signal since this is not expected when idle. Now in such cases local checkpoint handling is done in a manner that ensures that this node finishes in the proper state (`LCP_TAB_SAVED`). (Bug #30032863)

- Restoring tables for which `MAX_ROWS` was used to alter partitioning from a backup made from NDB 7.4 to a cluster running NDB 7.6 did not work correctly. This is fixed by ensuring that the upgrade code handling `PartitionBalance` supplies a valid table specification to the `NDB` dictionary. (Bug #29955656)

- During upgrade of an NDB Cluster when half of the data nodes were running NDB 7.6 while the remainder were running NDB 8.0, attempting to shut down those nodes which were running NDB 7.6 led to failure of one node with the error `CHECK FAILEDNODEPTR.P->DBLQHFAI`. (Bug #29912988, Bug #30141203)

- When performing a local checkpoint (LCP), a table's schema version was intermittently read as 0, which caused `NDB` LCP handling to treat the table as though it were being dropped. This could effect rebuilding of indexes offline by `ndb_restore` while the table was in the `TABLE_READ_ONLY` state. Now the function reading the schema version (`getCreateSchemaVersion()`) no longer not changes it while the table is read-only. (Bug #29910397)

- `NDB` index statistics are calculated based on the topology of one fragment of an ordered index; the fragment chosen in any particular index is decided at index creation time, both when the index is originally created, and when a node or system restart has recreated the index locally. This calculation is based in part on the number of fragments in the index, which can change when a table is reorganized. This means that, the next time that the node is restarted, this node may choose a different fragment, so that no fragments, one fragment, or two fragments are used to generate index statistics, resulting in errors from `ANALYZE TABLE`.

  This issue is solved by modifying the online table reorganization to recalculate the chosen fragment immediately, so that all nodes are aligned before and after any subsequent restart. (Bug #29534647)

- 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)

- The process of selecting the transaction coordinator checked for “live” data nodes but not necessarily for those that were actually available. (Bug #27160203)

Changes in MySQL NDB Cluster 7.6.11 (5.7.27-ndb-7.6.11) (2019-07-23)

MySQL NDB Cluster 7.6.11 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.27 (see Changes in MySQL 5.7.27 (2019-07-22)).

- 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. (WL #12303)

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. (WL #13117)

- **NDB Replication:** `NDB` did not handle binary logging of virtual generated columns of type `BLOB` correctly. Now such columns are always regarded as having zero length.

- A pushed join with `ORDER BY` did not always return the rows of the result in the specified order. This could occur when the optimizer used an ordered index to provide the ordering and the index used a column from the table that served as the root of the pushed join. (Bug #29860378)

- 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)

- Lack of `SharedGlobalMemory` was incorrectly reported as lack of undo buffer memory, even though the cluster used no disk data tables. (Bug #29806771)

  References: This issue is a regression of: Bug #92125, Bug #28537319.

- Long `TCKEYREQ` signals did not always use the expected format when invoked from `TCINDXREQ` processing. (Bug #29772731)

- Improved error message printed when the maximum offset for a `FIXED` column is exceeded. (Bug #29714670)

- Data nodes could fail due to an assert in the `DBTC` block under certain circumstances in resource-constrained environments. (Bug #29528188)

- An upgrade to NDB 7.6.9 or later from an earlier version could not be completed successfully if the redo log was filled to more than 25% of capacity. (Bug #29506844)

- When the `DBSPJ` block called the internal function `lookup_resume()` to schedule a previously enqueued operation, it used a correlation ID which could have been produced from its immediate ancestor in the execution order, and not its parent in the query tree as assumed. This could happen during execution of a `SELECT STRAIGHT_JOIN` query.

  Now `NDB` checks whether the execution ancestor is different from the query tree parent, and if not, performs a lookup of the query tree parent, and the parent's correlation ID is enqueued to be executed later. (Bug #29501263)

- When a new master took over, sending a `MASTER_LCP_REQ` signal and executing `MASTER_LCPCONF` from participating nodes, it expected that they had not completed the current local checkpoint under the previous master, which need not be true. (Bug #29487340, Bug #29601546)

- 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 7.6.10 (5.7.26-ndb-7.6.10) (2019-04-26)

MySQL NDB Cluster 7.6.10 is a new release of NDB 7.6, based on MySQL Server 5.7 and including features in version 7.6 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.26 (see Changes in MySQL 5.7.26 (2019-04-25)).

Bugs Fixed

- **NDB Disk Data:** The error message returned when validation of `MaxNoOfOpenFiles` in relation to `InitialNoOfOpenFiles` failed has been improved to make the nature of the problem clearer to users. (Bug #28943749)
- **NDB Disk Data:** Repeated execution of `ALTER TABLESPACE ... ADD DATAFILE` against the same tablespace caused data nodes to hang and left them, after being killed manually, unable to restart. (Bug #22605467)
- **NDB Cluster APIs:** `NDB` now identifies short-lived transactions not needing the reduction of lock contention provided by `NdbBlob::close()` and no longer invokes this method in cases (such as when

autocommit is enabled) in which unlocking merely causes extra work and round trips to be performed prior to committing or aborting the transaction. (Bug #29305592)

References: See also: Bug #49190, Bug #11757181.

- **NDB Cluster APIs:** When the most recently failed operation was released, the pointer to it held by `NdbTransaction` became invalid and when accessed led to failure of the NDB API application. (Bug #29275244)

- 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.

- API and data nodes running NDB 7.6 and later could not use an existing parsed configuration from an earlier release series due to being overly strict with regard to having values defined for configuration parameters new to the later release, which placed a restriction on possible upgrade paths. Now NDB 7.6 and later are less strict about having all new parameters specified explicitly in the configuration which they are served, and use hard-coded default values in such cases. (Bug #28993400)

- Added `DUMP 406` (`NdbfsDumpRequests`) to provide `NDB` file system information to global checkpoint and local checkpoint stall reports in the node logs. (Bug #28922609)

- A race condition between the `DBACC` and `DBLQH` kernel blocks occurred when different operations in a transaction on the same row were concurrently being prepared and aborted. This could result in `DBTUP` attempting to prepare an operation when a preceding operation had been aborted, which was unexpected and could thus lead to undefined behavior including potential data node failures. To solve this issue, `DBACC` and `DBLQH` now check that all dependencies are still valid before attempting to prepare an operation.

  Note

  This fix also supersedes a previous one made for a related issue which was originally reported as Bug #28500861.

  (Bug #28893633)

- The `ndbinfo.cpustat` table reported inaccurate information regarding send threads. (Bug #28884157)

- Execution of an LCP_COMPLETE_REP signal from the master while the LCP status was IDLE led to an assertion. (Bug #28871889)

- Issuing a `STOP` command in the `ndb_mgm` client caused `ndbmtd` processes which had recently been added to the cluster to hang in Phase 4 during shutdown. (Bug #28772867)

- 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)

- After a commit failed due to an error, `mysqld` shut down unexpectedly while trying to get the name of the table involved. This was due to an issue in the internal function `ndbcluster_print_error()`. (Bug #28435082)

- `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.

- Neither the `MAX_EXECUTION_TIME` optimizer hint nor the `max_execution_time` system variable was respected for DDL statements or queries against `INFORMATION_SCHEMA` tables while an `NDB` global schema lock was in effect. (Bug #27538139)

- When query memory was exhausted in the `DBSPJ` kernel block while storing correlation IDs for deferred operations, the query was aborted with error status 20000 `Query aborted due to out of query memory`. (Bug #26995027)

  References: See also: Bug #86537.

- `MaxBufferedEpochs` is used on data nodes to avoid excessive buffering of row changes due to lagging `NDB` event API subscribers; when epoch acknowledgements from one or more subscribers lag by this number of epochs, an asynchronous disconnection is triggered, allowing the data node to release the buffer space used for subscriptions. Since this disconnection is asynchronous, it may be the case that it has not completed before additional new epochs are completed on the data node, resulting in new epochs not being able to seize GCP completion records, generating warnings such as those shown here:

  ```
  [ndbd] ERROR    -- c_gcp_list.seize() failed...

  ...

  [ndbd] WARNING  -- ACK wo/ gcp record...
  ```

  And leading to the following warning:

  ```
  Disconnecting node %u because it has exceeded MaxBufferedEpochs
  (100 > 100), epoch ....
  ```

  This fix performs the following modifications:

  - Modifies the size of the GCP completion record pool to ensure that there is always some extra headroom to account for the asynchronous nature of the disconnect processing previously described, thus avoiding `c_gcp_list` seize failures.

- Modifies the wording of the `MaxBufferedEpochs` warning to avoid the contradictory phrase “100 > 100”.

(Bug #20344149)

- 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 7.6.9 (5.7.25-ndb-7.6.9) (2019-01-22)

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

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

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

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 5.7 through MySQL 5.7.25 (see Changes in MySQL 5.7.25 (2019-01-21)).

Bugs Fixed

- **Important Change:** When restoring to a cluster using data node IDs different from those in the original cluster, `ndb_restore` tried to open files corresponding to node ID 0. To keep this from happening, the `--nodeid` and `--backupid` options—neither of which has a default value—are both now explicitly required when invoking `ndb_restore`. (Bug #28813708)
- **Packaging; MySQL NDB ClusterJ:** `libndbclient` was missing from builds on some platforms. (Bug #28997603)
- **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:** A `DROP DATABASE` operation involving certain very large tables could lead to an unplanned shutdown of the cluster. (Bug #28855062)
- **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)
- **NDB Cluster APIs:** When the `NDB` kernel's `SUMA` block sends a `TE_ALTER` event, it does not keep track of when all fragments of the event are sent. When `NDB` receives the event, it buffers the fragments, and processes the event when all fragments have arrived. An issue could possibly arise for very large table definitions, when the time between transmission and reception could span multiple epochs; during this time, `SUMA` could send a `SUB_GCP_COMPLETE_REP` signal to indicate that it has sent all data for an

epoch, even though in this case that is not entirely true since there may be fragments of a TE_ALTER event still waiting on the data node to be sent. Reception of the SUB_GCP_COMPLETE_REP leads to closing the buffers for that epoch. Thus, when TE_ALTER finally arrives, NDB assumes that it is a duplicate from an earlier epoch, and silently discards it.

We fix the problem by making sure that the SUMA kernel block never sends a SUB_GCP_COMPLETE_REP for any epoch in which there are unsent fragments for a SUB_TABLE_DATA signal.

This issue could have an impact on NDB API applications making use of TE_ALTER events. (SQL nodes do not make any use of TE_ALTER events and so they and applications using them were not affected.) (Bug #28836474)

- Where a data node was restarted after a configuration change whose result was a decrease in the sum of MaxNoOfTables, MaxNoOfOrderedIndexes, and MaxNoOfUniqueHashIndexes, it sometimes failed with a misleading error message which suggested both a temporary error and a bug, neither of which was the case.

  The failure itself is expected, being due to the fact that there is at least one table object with an ID greater than the (new) sum of the parameters just mentioned, and that this table cannot be restored since the maximum value for the ID allowed is limited by that sum. The error message has been changed to reflect this, and now indicates that this is a permanent error due to a problem configuration. (Bug #28884880)

- 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.

- Running ANALYZE TABLE on an NDB table with an index having longer than the supported maximum length caused data nodes to fail. (Bug #28714864)

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

  References: See also: Bug #27622643.

- The output of ndb_config --configinfo --xml --query-all now shows that configuration changes for the ThreadConfig and MaxNoOfExecutionThreads data node parameters require system initial restarts (restart="system" initial="true"). (Bug #28494286)

- API nodes should observe that a node is moving through SL_STOPPING phases (graceful stop) and stop using the node for new transactions, which minimizes potential disruption in the later phases of the node shutdown process. API nodes were only informed of node state changes via periodic heartbeat signals, and so might not be able to avoid interacting with the node shutting down. This generated unnecessary failures when the heartbeat interval was long. Now when a data node is being gracefully stopped, all API nodes are notified directly, allowing them to experience minimal disruption. (Bug #28380808)

- Executing `SELECT * FROM INFORMATION_SCHEMA.TABLES` caused SQL nodes to restart in some cases. (Bug #27613173)
- When scanning a row using a `TUP` scan or `ACC` scan, or when performing a read using the primary key, it is possible to start a read of the row and hit a real-time break during which it is necessary to wait for the page to become available in memory. When the page request returns later, an attempt to read the row fails due to an invalid checksum; this is because, when the row is deleted, its checksum is invalidated.

  This problem is solved by introducing a new tuple header `DELETE_WAIT` flag, which is checked before starting any row scan or PK read operations on the row where disk data pages are not yet available, and cleared when the row is finally committed. (Bug #27584165, Bug #93035, Bug #28868412)
- 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)
- `mysqld` shut down unexpectedly when a purge of the binary log was requested before the server had completely started, and it was thus not yet ready to delete rows from the `ndb_binlog_index` table. Now when this occurs, requests for any needed purges of the `ndb_binlog_index` table are saved in a queue and held for execution when the server has completely started. (Bug #25817834)
- 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)
- `NdbScanFilter` did not always handle `NULL` according to the SQL standard, which could result in sending non-qualifying rows to be filtered (otherwise not necessary) by the MySQL server. (Bug #92407, Bug #28643463)

  References: See also: Bug #93977, Bug #29231709.
- `NDB` attempted to use condition pushdown on greater-than (`>`) and less-than (`<`) comparisons with `ENUM` column values but this could cause rows to be omitted in the result. Now such comparisons are no longer pushed down. Comparisons for equality (`=`) and inequality (`<>` / `!=`) with `ENUM` values are not affected by this change, and conditions including these comparisons can still be pushed down. (Bug #92321, Bug #28610217)

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

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

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

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

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

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

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

  (WL #11722)

Bugs Fixed

- **Packaging:** Expected NDB header files were in the `devel` RPM package instead of `libndbclient-devel`. (Bug #84580, Bug #26448330)
- **NDB Disk Data:** While restoring a local checkpoint, it is possible to insert a row that already exists in the database; this is expected behavior which is handled by deleting the existing row first, then inserting the new copy of that row. In some cases involving data on disk, `NDB` failed to delete the existing row. (Bug #91627, Bug #28341843)
- **NDB Client Programs:** Removed a memory leak in `NdbImportUtil::RangeList` that was revealed in ASAN builds. (Bug #91479, Bug #28264144)
- **MySQL NDB ClusterJ:** When a table containing a `BLOB` or a `TEXT` field was being queried with ClusterJ for a record that did not exist, an exception ("`The method is not valid in current blob state`") was thrown. (Bug #28536926)
- **MySQL NDB ClusterJ:** A `NullPointerException` was thrown when a full table scan was performed with ClusterJ on tables containing either a BLOB or a TEXT field. It was because the proper object initializations were omitted, and they have now been added by this fix. (Bug #28199372, Bug #91242)
- When copying deleted rows from a live node to a node just starting, it is possible for one or more of these rows to have a global checkpoint index equal to zero. If this happened at the same time that a full

local checkpoint was started due to the undo log getting full, the `LCP_SKIP` bit was set for a row having GCI = 0, leading to an unplanned shutdown of the data node. (Bug #28372628)

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

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

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

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

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

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

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

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

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

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

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

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

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

  References: See also: Bug #28893633.

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

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

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

- In some cases, a workload that included a high number of concurrent inserts caused data node failures when using debug builds. (Bug #91764, Bug #28387450, Bug #29055038)

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

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

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

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

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

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

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

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

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

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

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

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

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

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

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

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

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

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

  See What is New in NDB Cluster 7.6, as well as the descriptions of the parameters mentioned previously, for more information. (Bug #90709, Bug #27942974, Bug #27942583, WL #9638)

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

Bugs Fixed

- **ndbinfo Information Database:** It was possible following a restart for (sometimes incomplete) fallback data to be used in populating the `ndbinfo.processes` table, which could lead to rows in this table with empty `process_name` values. Such fallback data is no longer used for this purpose. (Bug #27985339)
- **NDB Client Programs:** The executable file `host_info` is no longer used by `ndb_setup.py`. This file, along with its parent directory `share/mcc/host_info`, has been removed from the NDB Cluster distribution.

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

  References: See also: Bug #27621546.

- **MySQL NDB ClusterJ:** ClusterJ could not be built from source using JDK 9. (Bug #27977985)
- An `NDB` restore operation failed under the following conditions:
  - A data node was restarted
  - The local checkpoint for the fragment being restored used two `.ctl` files
  - The first of these `.ctl` files was the file in use
  - The LCP in question consisted of more than 2002 parts

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

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

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

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

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

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

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

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

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

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

- The previously experimental shared memory (SHM) transporter is now supported in production. SHM works by transporting signals through writing them into memory, rather than on a socket. NDB already attempts to use SHM automatically between a data node and an API node sharing the same host. To enable explicit shared memory connections, set the `UseShm` configuration parameter to `1` for the relevant data node. When explicitly defining shared memory as the connection method, it is also necessary that the data node is identified by `HostName` and the API node by `HostName`.

  Additional tuning parameters such as `ShmSize`, `ShmSpintime`, and `SendBufferMemory` can be employed to improve performance of the SHM transporter. Configuration of SHM is otherwise similar to that of the TCP transporter. The `SigNum` parameter is no longer used, and any settings made for it are now ignored. NDB Cluster Shared Memory Connections, provides more information about these parameters.

  In addition, as part of this work, `NDB` code relating to support for the legacy SCI transporter, which had long been unsupported, has been removed. See `www.dolphinics.com` for information about support for legacy SCI hardware or information about the newer Dolphin Express hardware. (WL #7512)

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

  Note

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

  (WL #11164)

- The poll receiver which `NDB` uses to read from sockets, execute messages from the sockets, and wake up other threads now offloads wakeup of other threads to a new thread that wakes up the other threads on request, and otherwise simply sleeps. This improves the scalability of a single cluster connection by keeping the receive thread from becoming overburdened by tasks including wakeup of other threads. (WL #9663)

Bugs Fixed

- **Important Change; NDB Client Programs:** `ndb_top` ignored short forms of command-line options, and did not in all cases handle misformed long options correctly. As part of the fix for these issues, the following changes have been made to command-line options used with `ndb_top` to bring them more into line with those used with other NDB Cluster and MySQL programs:

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

  (Bug #26907833)

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

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

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

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

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

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

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

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

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

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

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

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

  At restore time, the log is replayed up to the stop GCI, restoring the system to the state it had at the consistent stop GCI. A problem arose when, under load, it was possible to select a GCI boundary which

occurred too early and did not span all the data captured. This could lead to inconsistencies when restoring the backup; these could be noticed as broken constraints or corrupted `BLOB` entries.

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

References: See also: Bug #27566346.

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

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

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

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

  References: See also: Bug #28728603.

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

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

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

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

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

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

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

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

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

  References: See also: Bug #20112700.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Bugs Fixed

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

- **NDB Client Programs:** The Auto-Installer did not provide a mechanism for setting the `ServerPort` parameter. (Bug #89623, Bug #27539823)

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

  References: See also: Bug #28698831.

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

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

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

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

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

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

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

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

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

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

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

- **Important Change; NDB Disk Data:** NDB Cluster has improved node restart times and overall performance with larger data sets by implementing partial local checkpoints (LCPs). Prior to this release, an LCP always made a copy of the entire database.

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

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

  Important

  Upgrading to NDB 7.6.4 or downgrading from this release requires purging then re-creating the `NDB` data node file system, which means that an initial restart of

> each data node is needed. An initial node restart still requires a complete LCP; a partial LCP is not used for this purpose.
>
> A rolling restart or system restart is a normal part of an `NDB` software upgrade. When such a restart is performed as part of an upgrade to NDB 7.6.4 or later, any existing LCP files are checked for the presence of the LCP `sysfile`, indicating that the existing data node file system was written using NDB 7.6.4 or later. If such a node file system exists, but does not contain the `sysfile`, and if any data nodes are restarted without the `--initial` option, `NDB` causes the restart to fail with an appropriate error message. This detection can be performed only as part of an upgrade; it is not possible to do so as part of a downgrade to NDB 7.6.3 or earlier from a later release.
>
> *Exception*: If there are no data node files—that is, in the event of a "clean" start or restart—using `--initial` is not required for a software upgrade, since this is already equivalent to an initial restart. (This aspect of restarts is unchanged from previous releases of NDB Cluster.)

In addition, the default value for `StartPartitionedTimeout` is changed from 60000 to 0.

This release also deprecates the data node configuration parameters `BackupDataBufferSize`, `BackupWriteSize`, and `BackupMaxWriteSize`; these are now subject to removal in a future NDB Cluster version. (Bug #27308632, WL #8069, WL #10302, WL #10993)

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

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

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

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

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

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

  See The NDB Cluster Auto-Installer (NDB 7.5) (NO LONGER SUPPORTED), for more information. (WL #10340, WL #10408, WL #10449)

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

  - Added a mechanism for setting SSH membership on a per-host basis.
  - Updated the Paramiko Python module to the most recent available version (2.6.1).
  - Provided a place in the GUI for encrypted private key passwords, and discontinued use of hardcoded passwords.

Related enhancements implemented in the current release include the following:

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

See The NDB Cluster Auto-Installer (NDB 7.5) (NO LONGER SUPPORTED), for more information. (WL #10426, WL #11128, WL #11289)

- **MySQL NDB ClusterJ:** ClusterJ now supports CPU binding for receive threads through the setRecvThreadCPUids() and getRecvThreadCPUids() methods. Also, the receive thread activation threshold can be set and get with the setRecvThreadActivationThreshold() and getRecvThreadActivationThreshold() methods. (WL #10815)
- It is now possible to specify a set of cores to be used for I/O threads performing offline multithreaded builds of ordered indexes, as opposed to normal I/O duties such as file I/O , compression , or decompression. “Offline” in this context refers to building of ordered indexes performed when the parent table is not being written to; such building takes place when an `NDB` cluster performs a node or system restart, or as part of restoring a cluster from backup using `ndb_restore --rebuild-indexes`.

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

  This enhancement is implemented as follows:

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

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

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

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

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

Note

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

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

(Bug #25428560)

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

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

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

- `ThreadConfig` now has an additional `nosend` parameter that can be used to prevent a `main`, `ldm`, `rep`, or `tc` thread from assisting the send threads, by setting this parameter to 1 for the given thread. By default, `nosend` is 0. It cannot be used with threads other than those of the types just listed. (WL #11554)

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

  Since the effect of this change is generally to require fewer SPJ requests and instances, performance of pushed-down joins should be improved in many cases. (WL #10234)

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

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

- `MaxUIBuildBatchSize`: Maximum scan batch size used for building unique keys.

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

Increasing the appropriate batch size or sizes can help amortize inter-thread and inter-node latencies and make use of more parallel resources (local and remote) to help scale DDL performance. (WL #11158)

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

  There are no user-visible changes in functionality directly associated with this work; this performance enhancement is part of the work being done in NDB 7.6 to improve undo long handling for partial local checkpoints. (WL #8478)

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

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

  Mapping from page ID to table ID and fragment ID is now done using information the extent header contains about the table IDs and fragment IDs of the pages used in a given extent. Since the extent pages are always present in the page cache, no extra disk reads are required to perform the mapping, and the information can be read using existing `TSMAN` data structures. (WL #10194)

- Added the `NODELOG DEBUG` command in the `ndb_mgm` client to provide runtime control over data node debug logging. `NODE DEBUG ON` causes a data node to write extra debugging information to its node log, the same as if the node had been started with `--verbose`. `NODELOG DEBUG OFF` disables the extra logging. (WL #11216)

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

  - If requested data is not found on the same node, reads can be directed to another node in the same availability domain.

  - Communication between nodes in different availability domains are guaranteed to use `NDB` transporters' WAN support without any further manual intervention.

  - The transporter's group number can be based on which availability domain is used, such that also SQL and other API nodes communicate with local data nodes in the same availability domain whenever possible.

  - The arbitrator can be selected from an availability domain in which no data nodes are present, or, if no such availability domain can be found, from a third availability domain.

  This parameter takes an integer value between 0 and 16, with 0 being the default; using 0 is the same as leaving `LocationDomainId` unset. (WL #10172)

Bugs Fixed

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  - A new error code is added to assist testing.

  (Bug #26364729)

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

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

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

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

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

  References: See also: Bug #27067538.

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

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

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

- When representing a materialized 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)

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

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

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

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

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

  References: See also: Bug #20981491.

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

  This was is a side effect of a fix for a previous issue, which disabled CPU locking based on the version of the available `glibc`. The specific `glibc` issue being guarded against is encountered only in response to an internal NDB API call (`Ndb_UnlockCPU()`) not used by data nodes (and which can be accessed

only through internal API calls). The current fix enables CPU locking for data nodes and disables it only for the relevant API calls when an affected `glibc` version is used. (Bug #87683, Bug #26758939)

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

- `ndb_top` failed to build on platforms where the `ncurses` library did not define `stdscr`. Now these platforms require the `tinfo` library to be included. (Bug #87185, Bug #26524441)
- On completion of a local checkpoint, every node sends a `LCP_COMPLETE_REP` signal to every other node in the cluster; a node does not consider the LCP complete until it has been notified that all other nodes have sent this signal. Due to a minor flaw in the LCP protocol, if this message was delayed from another node other than the master, it was possible to start the next LCP before one or more nodes had completed the one ongoing; this caused problems with `LCP_COMPLETE_REP` signals from previous LCPs becoming mixed up with such signals from the current LCP, which in turn led to node failures.

  To fix this problem, we now ensure that the previous LCP is complete before responding to any `TCGETOPSIZEREQ` signal initiating a new LCP. (Bug #87184, Bug #26524096)
- NDB Cluster did not compile successfully when the build used `WITH_UNIT_TESTS=OFF`. (Bug #86881, Bug #26375985)
- Recent improvements in local checkpoint handling that use `OM_CREATE` to open files did not work correctly on Windows platforms, where the system tried to create a new file and failed if it already existed. (Bug #86776, Bug #26321303)
- A potential hundredfold signal fan-out when sending a `START_FRAG_REQ` signal could lead to a node failure due to a `job buffer full` error in start phase 5 while trying to perform a local checkpoint during a restart. (Bug #86675, Bug #26263397)

  References: See also: Bug #26288247, Bug #26279522.
- Compilation of NDB Cluster failed when using `-DWITHOUT_SERVER=1` to build only the client libraries. (Bug #85524, Bug #25741111)
- The `NDBFS` block's `OM_SYNC` flag is intended to make sure that all FSWRITEREQ signals used for a given file are synchronized, but was ignored by platforms that do not support `O_SYNC`, meaning that this feature did not behave properly on those platforms. Now the synchronization flag is used on those platforms that do not support `O_SYNC`. (Bug #76975, Bug #21049554)

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

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

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

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

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

- Packaging Notes

- Functionality Added or Changed
- Bugs Fixed

Packaging Notes

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

Functionality Added or Changed

- **Important Change; MySQL NDB ClusterJ:** The ClusterJPA plugin for OpenJPA is no longer supported by NDB Cluster, and has been removed from the distribution. (Bug #23563810)
- **NDB Replication:** Added the `--ndb-log-update-minimal` option for logging by `mysqld`. This option causes only primary key values to be written in the before image, and only changed columns in the after image. (Bug #24438868)
- **MySQL NDB ClusterJ:** A new automatic reconnection feature has been implemented to facilitate the handling of connectivity issues. The feature is enabled by setting a positive number for a new connection property, `com.mysql.clusterj.connection.autoreconnect.timeout`, which specifies the length of the timeout period in seconds. If a connectivity error occurs, ClusterJ attempts to reconnect the application to the NDB Cluster after the application closes all the sessions; if the application does not close all sessions within the timeout period, ClusterJ closes any open sections forcibly, and then attempts reconnection. See Error Handling and Reconnection for details. (WL #9545)
- In some critical situations such as data node failure, it was possible for the volume of log messages produced to cause file system and other issues, which compounded the problem, due to the fact that these messages were logged synchronously using `stdout`. To keep this from happening, log messages from worker threads now use a log buffer instead, which is nonblocking, and thus much less likely to cause interference with other processes under such conditions. (Bug #24748843)
- Added the `--diff-default` option for `ndb_config`. This option causes the program to print only those parameters having values that differ from their defaults. (Bug #85831, Bug #25844166)
- Added the `ndb_top` program on unix-based platforms. This utility shows CPU load and usage information for an `NDB` data node, with periodic updates (each second, by default). The display is in text or color ASCII graph format; both formats can be displayed at the same time. It is also possible to disable color output for the graph.

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

  `ndb_top` is not currently available for Windows platforms.

  For more information, see ndb_top — View CPU usage information for NDB threads. (WL #9788)

Bugs Fixed

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

  (Bug #85679, Bug #25799465)

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

- **NDB Replication:** Added a check to stop an `NDB` replication slave when configuration as a multithreaded slave is detected (for example, if `slave_parallel_workers` is set to a nonzero value). (Bug #21074209)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  References: See also: Bug #16275684.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

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

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

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

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

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

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

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

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

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

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

    The `RESERVED` resource has been removed.

  - `IndexMemory` is no longer displayed in `ndb_config` output.

  (WL #9835, WL #10196)

- **Performance:** A number of debugging statements and printouts in the sources for the `DBTC` and `DBLQH` kernel blocks, as well as in related code, were moved into debugging code or removed altogether. This is expected to result in an improvement of up to 10% in the performance of local data management and transaction coordinator threads in many common use cases. (WL #10188)

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

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

  For more information, see The ndbinfo config_nodes Table, and The ndbinfo processes Table, as well as Ndb_cluster_connection::set_service_uri(). (WL #9819, WL #10147)

- **NDB Cluster APIs:** The system name of an NDB cluster is now visible in the `mysql` client as the value of the `Ndb_system_name` status variable, and can also be obtained by NDB API application using the `Ndb_cluster_connection::get_system_name()` method. The system name can be set using the `Name` parameter in the `[system]` section of the cluster configuration file. (WL #10321)

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

- Previously, when one LDM thread experienced I/O lag, such as during a disk overload condition, it wrote to a local checkpoint more slowly—that is, it wrote in I/O lag mode. However, other LDM threads did not necessarily observe or conform to this state. To ensure that write speed for the LCP is reduced by all LDM threads when such a slowdown is encountered, `NDB` now tracks I/O lag mode globally, so that I/O lag state is reported as soon as at least one thread is writing in I/O lag mode, and thus all LDM threads are forced to write in lag mode while the lag condition persists. This reduction in write speed by other LDM instances should increase overall capacity, enabling the disk overload condition to be overcome more quickly in such cases than before. (WL #10174)

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

  For more information, see ndb_import — Import CSV Data Into NDB. (WL #7614, WL #8862, WL #10653)

Bugs Fixed

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

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

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

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

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

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

- **Solaris:** The minimum required version of Solaris is now Solaris 11 update 3, due to a dependency on system runtime libraries.

- **Solaris:** On Solaris, MySQL is now built with Developer Studio 12.5 instead of `gcc`. The binaries require the Developer Studio C/C++ runtime libraries to be installed. See here for how to install only the libraries:

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

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

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

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

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

the event buffer had unlimited memory (`ndb_eventbuffer_max_alloc` = 0, the default), which could confuse users.

This is fixed as follows:

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

(Bug #25726723)

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

  References: See also: Bug #20092754.

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

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

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

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

  References: See also: Bug #25306089.

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

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

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

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

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

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

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

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

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

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

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

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

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

  References: See also: Bug #24926009.

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

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

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

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

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

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

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

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

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

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

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

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

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

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

  For more information, see Upgrading and Downgrading NDB Cluster. (WL #9778)

Bugs Fixed

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

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

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

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

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

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

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

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

  References: See also: Bug #25182956.

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

  References: See also: Bug #25302901.

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

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

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

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

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

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

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

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

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

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

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

  - Wait for data nodes to start shutting down (maximum 12 seconds)
  - Wait for data nodes to complete shutting down and reach `NOT_STARTED` state (maximum 120 seconds)

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

  References: See also: Bug #28728485.

Release Series Changelogs: MySQL NDB Cluster 7.6

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

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

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

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

Changes in MySQL NDB Cluster 7.6.30 (5.7.44-ndb-7.6.30) (2024-04-30)

Bugs Fixed

- **NDB Client Programs:** `ndb_redo_log_reader` exited with `Record type = 0 not implemented` when reaching an unused page, all zero bytes, or a page which was only partially used (typically a page consisting of the page header only). (Bug #36313259)
- Repeated incomplete incomplete attempts to perform a system restart in some cases left the cluster in a state from which it could not recover without restoring it from backup. (Bug #35801548)
- The event buffer used by the NDB API maintains an internal pool of free memory to reduce the interactions with the runtime and operating system, while allowing memory that is no longer needed to be returned for other uses. This free memory is subtracted from the total allocated memory to determine the memory is use which is reported and used for enforcing buffer limits and other purposes; this was represented using a 32-bit value, so that if it exceeded 4 GB, the value wrapped, and the amount of free memory appeared to be reduced. This had potentially adverse effects on event buffer memory release to the runtime and OS, free memory reporting, and memory limit handling.

  This is fixed by using a 64-bit value to represent the amount of pooled free memory. (Bug #35483764)

  References: See also: Bug #35655162, Bug #35663761.
- Removed unnecessary warnings generated by transient disconnections of data nodes during restore operations. (Bug #33144487)
- In some cases, when trying to perform an online add index operation on an `NDB` table with no explicit primary key (see Limitations of NDB online operations), the resulting error message did not make the nature of the problem clear. (Bug #30766579)

  References: See also: Bug #36382071.

Changes in MySQL NDB Cluster 7.6.29 (5.7.44-ndb-7.6.29) (2024-01-16)

- Compilation Notes
- Bugs Fixed

Compilation Notes

- **Microsoft Windows:** NDB Cluster did not compile correctly using Visual Studio 2022. (Bug #35967676)
- NDB Cluster did not compile correctly on Ubuntu 23.10. (Bug #35847193)

Bugs Fixed

- When a node failure is detected, transaction coordinator (TC) instances check their own transactions to determine whether they need handling to ensure completion, implemented by checking whether each transaction involves the failed node, and if so, marking it for immediate timeout handling. This causes the transaction to be either rolled forward (commit) or back (abort), depending on whether it had started committing, using the serial commit protocol. When the TC was in the process of getting permission to commit (`CS_PREPARE_TO_COMMIT`), sending commit requests (`CS_COMMITTING`), or sending completion requests (`CS_COMPLETING`), timeout handling waited until the transaction was in a stable state before commencing the serial commit protocol.

  Prior to the fix for Bug#22602898, all timeouts during `CS_COMPLETING` or `CS_COMMITTING` resulted in switching to the serial commit-complete protocol, so skipping the handling in any of the three states cited

previously did not stop the prompt handling of the node failure. It was found later that this fix removed the blanket use of the serial commit-complete protocol for commit-complete timeouts, so that when handling for these states was skipped, no node failure handling action was taken, with the result that such transactions hung in a commit or complete phase, blocking checkpoints.

The fix for Bug#22602898 removed this stable state handling to avoid it accidentally triggering, but this change also stopped it from triggering when needed in this case where node failure handling found a transaction in a transient state. We solve this problem by modifying `CS_COMMIT_SENT` and `CS_COMPLETE_SENT` stable state handling to perform node failure processing if a timeout has occurred for a transaction with a failure number different from the current latest failure number, ensuring that all transactions involving the failed node are in fact eventually handled. (Bug #36028828)

References: See also: Bug #22602898.

- It was possible for the `readln_socket()` function in `storage/ndb/src/common/util/socket_io.cpp` to read one character too many from the buffer passed to it as an argument. (Bug #35857936)

- The slow disconnection of a data node while a management server was unavailable could sometimes interfere with the rolling restart process. This became especially apparent when the cluster was hosted by NDB Operator, and the old `mgmd` pod did not recognize the IP address change of the restarted data node pod; this was visible as discrepancies in the output of `SHOW STATUS` on different management nodes.

  We fix this by making sure to clear any cached address when connecting to a data node so that the data node's new address (if any) is used instead. (Bug #35667611)

- The maximum permissible value for the oldest restorable global checkpoint ID is `MAX_INT32` (4294967295). Such an ID greater than this value causes the data node to shut down, requiring a backup and restore on a cluster started with `--initial`.

  Now, approximately 90 days before this limit is reached under normal usage, an appropriate warning is issued, allowing time to plan the required corrective action. (Bug #35641420)

  References: See also: Bug #35749589.

- Subscription reports were sent out too early by `SUMA` during a node restart, which could lead to schema inconsistencies between cluster SQL nodes. In addition, an issue with the `ndbinfo restart_info` table meant that restart phases for nodes that did not belong to any node group were not always reported correctly. (Bug #30930132)

- Online table reorganization inserts rows from existing table fragments into new table fragments; then, after committing the inserted rows, it deletes the original rows. It was found that the inserts caused `SUMA` triggers to fire, and binary logging to occur, which led to the following issues:

  - Inconsistent behavior, since DDL is generally logged as one or more statements, if at all, rather than by row-level effect.
  - It was incorrect, since only writes were logged, but not deletes.
  - It was unsafe since tables with blobs did not receive associated the row changes required to form valid binary log events.
  - It used CPU and other resources needlessly.

  For tables with no blob columns, this was primarily a performance issue; for tables having blob columns, it was possible for this behavior to result in unplanned shutdowns of mysqld processes performing binary logging and perhaps even data corruption downstream. (Bug #19912988)

  References: See also: Bug #16028096, Bug #34843617.

Changes in MySQL NDB Cluster 7.6.28 (5.7.44-ndb-7.6.28) (2023-10-26)

Bugs Fixed

- **NDB Cluster APIs:** `Ndb::pollEvents2()` did not set **`NDB_FAILURE_GCI`** (`~(Uint64)0`) to indicate cluster failure. (Bug #35671818)

  References: See also: Bug #31926584. This issue is a regression of: Bug #18753887.

- **NDB Client Programs:** When `ndb_select_all` failed to read all data from the table, it always tried to re-read it. This could lead to the two problems listed here:

  - Returning a non-empty partial result eventually led to spurious reports of duplicate rows.
  - The table header was printed on every retry.

  Now when `ndb_select_all` is unsuccessful at reading all the table data, its behavior is as follows:

  - When the result is non-empty, `ndb_select_all` halts with an error (and does not retry the scan of the table).
  - When the result is empty, `ndb_select_all` retries the scan, reusing the old header.

  (Bug #35510814)

- Following a node connection failure, the transporter registry's error state was not cleared before initiating a reconnect, which meant that the error causing the connection to be disconnected originally might still be set; this was interpreted as a failure to reconnect. (Bug #35774109)

- When a `TransporterRegistry` (TR) instance connects to a management server, it first uses the MGM API, and then converts the connection to a `Transporter` connection for further communication. The initial connection had an excessively long timeout (60 seconds) so that, in the case of a cluster having two management servers where one was unavailable, clients were forced to wait until this management server timed out before being able to connect to the available one.

  We fix this by setting the MGM API connection timeout to 5000 milliseconds, which is equal to the timeout used by the TR for getting and setting dynamic ports. (Bug #35714466)

- Values for causes of conflicts used in conflict resolution exceptions tables were misaligned such that the order of `ROW_ALREADY_EXISTS` and `ROW_DOES_NOT_EXIST` was reversed. (Bug #35708719)

- In cases where the distributed global checkpoint (GCP) protocol stops making progress, this is detected and optionally handled by the GCP monitor, with handling as determined by the `TimeBetweenEpochsTimeout` and `TimeBetweenGlobalCheckpointsTimeout` data node parameters.

  The LCP protocol is mostly node-local, but depends on the progress of the GCP protocol at the end of a local checkpoint (LCP); this means that, if the GCP protocol stalls, LCPs may also stall in this state. If the LCP watchdog detects that the LCP is stalled in this end state, it should defer to the GCP monitor to handle this situation, since the GCP Monitor is distribution-aware.

  If no GCP monitor limit is set (`TimeBetweenEpochsTimeout` is equal 0), no handling of GCP stalls is performed by the GCP monitor. In this case, the LCP watchdog was still taking action which could eventually lead to cluster failure; this fix corrects this misbehavior so that the LCP watchdog no longer takes any such action. (Bug #29885899)

- Previously, when a timeout was detected during transaction commit and completion, the transaction coordinator (TC) switched to a serial commit-complete execution protocol, which slowed commit-complete processing for large transactions, affecting `GCP_COMMIT` delays and epoch sizes. Instead of switching in such cases, the TC now continues waiting for parallel commit-complete, periodically logging a transaction summary, with states and nodes involved. (Bug #22602898)

  References: See also: Bug #35260944.

Changes in MySQL NDB Cluster 7.6.27 (5.7.43-ndb-7.6.27) (2023-07-18)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change; NDB Cluster APIs:** The `NdbRecord` interface allows equal changes of primary key values; that is, you can update a primary key value to its current value, or to a value which compares as equal according to the collation rules being used, without raising an error. `NdbRecord` does not itself try to prevent the update; instead, the data nodes check whether a primary key is updated to an unequal value and in this case reject the update with Error 897: `Update attempt of primary key via ndbcluster internal api`.

  Previously, when using any other mechanism than `NdbRecord` in an attempt to update a primary key value, the NDB API returned error 4202 `Set value on tuple key attribute is not allowed`, even setting a value identical to the existing one. With this release, the check when performing updates by other means is now passed off to the data nodes, as it is already by `NdbRecord`.

  This change applies to performing primary key updates with `NdbOperation::setValue()`, `NdbInterpretedCode::write_attr()`, and other methods of these two classes which set column values (including `NdbOperation` methods `incValue()`, `subValue()`, `NdbInterpretedCode` methods `add_val()`, `sub_val()`, and so on), as well as the `OperationOptions::OO_SETVALUE` extension to the `NdbOperation` interface. (Bug #35106292)

- NDB 7.6 is now built with support for OpenSSL 3.0. (WL #15614)

Bugs Fixed

- Backups using `NOWAIT` did not start following a restart of the data node. (Bug #35389533)

- When handling the connection (or reconnection) of an API node, it was possible for data nodes to inform the API node that it was permitted to send requests too quickly, which could result in requests not being delivered and subsequently timing out on the API node with errors such as Error 4008 `Receive from Ndb failed` or Error 4012 `Request ndbd time-out, maybe due to high load or communication problems`. (Bug #35387076)

- Made the following improvements in warning output:

  - Now, in addition to local checkpoint (LCP) elapsed time, the maximum time allowed without any progress is also printed.

  - Table IDs and fragment IDs are undefined and thus not relevant when an LCP has reached `WAIT_END_LCP` state, and are no longer printed at that point.

  - When the maximum limit was reached, the same information was shown twice, as both warning and crash information.

  (Bug #35376705)

- When deferred triggers remained pending for an uncommitted transaction, a subsequent transaction could waste resources performing unnecessary checks for deferred triggers; this could lead to an unplanned shutdown of the data node if the latter transaction had no committable operations.

  This was because, in some cases, the control state was not reinitialized for management objects used by `DBTC`.

  We fix this by making sure that state initialization is performed for any such object before it is used. (Bug #35256375)

- A pushdown join between queries featuring very large and possibly overlapping `IN()` and `NOT IN()` lists caused SQL nodes to exit unexpectedly. One or more of the `IN()` (or `NOT IN()`) operators required in excess of 2500 arguments to trigger this issue. (Bug #35185670, Bug #35293781)

- The buffers allocated for a key of size `MAX_KEY_SIZE` were of insufficient size. (Bug #35155005)

- Some calls made by the `ndbcluster` handler to `push_warning_printf()` used severity level `ERROR`, which caused an assertion in debug builds. This fix changes all such calls to use severity `WARNING` instead. (Bug #35092279)

- When a connection between a data node and an API or management node was established but communication was available only from the other node to the data node, the data node considered the other node “live”, since it was receiving heartbeats, but the other node did not monitor heartbeats and so reported no problems with the connection. This meant that the data node assumed wrongly that the other node was (fully) connected.

  We solve this issue by having the API or management node side begin to monitor data node liveness even before receiving the first `REGCONF` signal from it; the other node sends a `REGREQ` signal every 100 milliseconds, and only if it receives no `REGCONF` from the data node in response within 60 seconds is the node reported as disconnected. (Bug #35031303)

- The log contained a high volume of messages having the form `DICT: index` *`index number`* `stats auto-update requested`, logged by the `DBDICT` block each time it received a report from `DBTUX` requesting an update. These requests often occur in quick succession during writes to the table, with

the additional possibility in this case that duplicate requests for updates to the same index were being logged.

Now we log such messages just before `DBDICT` actually performs the calculation. This removes duplicate messages and spaces out messages related to different indexes. Additional debug log messages are also introduced by this fix, to improve visibility of the decisions taken and calculations performed. (Bug #34760437)

- Local checkpoints (LCPs) wait for a global checkpoint (GCP) to finish for a fixed time during the end phase, so they were performed sometimes even before all nodes were started.

  In addition, this bound, calculated by the GCP coordinator, was available only on the coordinator itself, and only when the node had been started (start phase 101).

  These two issues are fixed by calculating the bound earlier in start phase 4; GCP participants also calculate the bound whenever a node joins or leaves the cluster. (Bug #32528899)

Changes in MySQL NDB Cluster 7.6.26 (5.7.42-ndb-7.6.26) (2023-04-19)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **MySQL NDB ClusterJ:** Performance has been improved for accessing tables using a single-column partition key when the column is of type `CHAR` or `VARCHAR`. (Bug #35027961)
- Beginning with this release, `ndb_restore` implements the `--timestamp-printouts` option, which causes all error, info, and debug node log messages to be prefixed with timestamps. (Bug #34110068)

Bugs Fixed

- **Microsoft Windows:** Two memory leaks found by code inspection were removed from `NDB` process handles on Windows platforms. (Bug #34872901)
- **Microsoft Windows:** On Windows platforms, the data node angel process did not detect whether a child data node process exited normally. We fix this by keeping an open process handle to the child and using this when probing for the child's exit. (Bug #34853213)
- **NDB Cluster APIs; MySQL NDB ClusterJ:** MySQL ClusterJ uses a scratch buffer for primary key hash calculations which was limited to 10000 bytes, which proved too small in some cases. Now we `malloc()` the buffer if its size is not sufficient.

  This also fixes an issue with the `Ndb` object methods `startTransaction()` and `computeHash()` in the NDB API: Previously, if either of these methods was passed a temporary buffer of insufficient size, the method failed. Now in such cases a temporary buffer is allocated.

  Our thanks to Mikael Ronström for this contribution. (Bug #103814, Bug #32959894)
- **NDB Cluster APIs:** When dropping an event operation (`NdbEventOperation`) in the NDB API, it was sometimes possible for the dropped event operation to remain visible to the application after instructing the data nodes to stop sending events related to this event operation, but before all pending buffered events were consumed and discarded. This could be observed in certain cases when performing an online alter operation, such as `ADD COLUMN` or `RENAME COLUMN`, along with concurrent writes to the affected table.

Further analysis showed that the dropped events were accessible when iterating through event operations with `Ndb::getGCIEventOperations()`. Now, this method skips dropped events when called iteratively. (Bug #34809944)

- To reduce confusion between the version of the file format and the version of the cluster which produced the backup, the backup file format version is now shown by `ndb_restore` using hexadecimal notation. (Bug #35079426)

  References: This issue is a regression of: Bug #34110068.

- Removed a memory leak in the `DBDICT` kernel block caused when an internal foreign key definition record was not released when no longer needed. This could be triggered by either of the following events:

  - Drop of a foreign key constraint on an `NDB` table
  - Rejection of an attempt to create a foreign key constraint on an `NDB` table

  Such records use the `DISK_RECORDS` memory resource; you can check this on a running cluster by executing `SELECT node_id, used FROM ndbinfo.resources WHERE resource_name='DISK_RECORDS'` in the `mysql` client. This resource uses `SharedGlobalMemory`, exhaustion of which could lead not only to the rejection of attempts to create foreign keys, but of queries making use of joins as well, since the `DBSPJ` block also uses shared global memory by way of `QUERY_MEMORY`. (Bug #35064142)

- When a transaction coordinator is starting fragment scans with many fragments to scan, it may take a realtime break (RTB) during the process to ensure fair CPU access for other requests. When the requesting API disconnected and API failure handling for the scan state occurred before the RTB continuation returned, continuation processing could not proceed because the scan state had been removed.

  We fix this by adding appropriate checks on the scan state as part of the continuation process. (Bug #35037683)

- Sender and receiver signal IDs were printed in trace logs as signed values even though they are actually unsigned 32-bit numbers. This could result in confusion when the top bit was set, as it cuased such numbers to be shown as negatives, counting upwards from `-MAX_32_BIT_SIGNED_INT`. (Bug #35037396)

- A fiber used by the `DICT` block monitors all indexes, and triggers index statistics calculations if requested by `DBTUX` index fragment monitoring; these calculations are performed using a schema transaction. When the `DICT` fiber attempts but fails to seize a transaction handle for requesting a schema transaction to be started, fiber exited, so that no more automated index statistics updates could be performed without a node failure. (Bug #34992370)

  References: See also: Bug #34007422.

- Schema objects in NDB use composite versioning, comprising major and minor subversions. When a schema object is first created, its major and minor versions are set; when an existing schema object is altered in place, its minor subversion is incremented.

  At restart time each data node checks schema objects as part of recovery; for foreign key objects, the versions of referenced parent and child tables (and indexes, for foreign key references not to or from a table's primary key) are checked for consistency. The table version of this check compares only major subversions, allowing tables to evolve, but the index version also compares minor subversions; this resulted in a failure at restart time when an index had been altered.

We fix this by comparing only major subversions for indexes in such cases. (Bug #34976028)

References: See also: Bug #21363253.

- When running an NDB Cluster with multiple management servers, termination of the `ndb_mgmd` processes required an excessive amount of time when shutting down the cluster. (Bug #34872372)
- When requesting a new global checkpoint (GCP) from the data nodes, such as by the NDB Cluster handler in `mysqld` to speed up delivery of schema distribution events and responses, the request was sent 100 times. While the `DBDIH` block attempted to merge these duplicate requests into one, it was possible on occasion to trigger more than one immediate GCP. (Bug #34836471)
- After receiving a `SIGTERM` signal, `ndb_mgmd` did not wait for all threads to shut down before exiting. (Bug #33522783)

  References: See also: Bug #32446105.
- When started with no connection string on the command line, `ndb_waiter` printed `Connecting to mgmsrv at (null)`. Now in such cases, it prints `Connecting to management server at nodeid=0, localhost:1186` if no other default host is specified.

  The `--help` option and other `ndb_waiter` program output was also improved. (Bug #12380163)

Changes in MySQL NDB Cluster 7.6.25 (5.7.41-ndb-7.6.25) (2023-01-18)

Bugs Fixed

- **MySQL NDB ClusterJ:** ClusterJ could not be built on Ubuntu 22.10 with GCC 12.2. (Bug #34666985)
- In some contexts, a data node process may be sent `SIGCHLD` by other processes. Previously, the data node process bound a signal handler treating this signal as an error, which could cause the process to shut down unexpectedly when run in the foreground in a Kubernetes environment (and possibly under other conditions as well). This occurred despite the fact that a data node process never starts child processes itself, and thus there is no need to take action in such cases.

  To fix this, the handler has been modified to use `SIG_IGN`, which should result in cleanup of any child processes.

  Note

  `mysqld` and `ndb_mgmd` processes do not bind any handlers for `SIGCHLD`.

  (Bug #34826194)
- The running node from a node group scans each fragment (`CopyFrag`) and sends the rows to the starting peer in order to synchronize it. If a row from the fragment is locked exclusively by a user transaction, it blocks the scan from reading the fragment, causing the copyFrag to stall.

  If the starting node fails during the `CopyFrag` phase then normal node failure handling takes place. The cordinator node's transaction coordinator (TC) performs TC takeover of the user transactions from the TCs on the failed node. Since the scan that aids copying the fragment data over to the starting node is considered internal only, it is not a candidate for takeover, thus the takeover TC marks the `CopyFrag` scan as closed at the next opportunity, and waits until it is closed.

  The current issue arose when the `CopyFrag` scan was in the `waiting for row lock` state, and the closing of the marked scan was not performed. This led to TC takeover stalling while waiting for the

close, causing unfinished node failure handling, and eventually a GCP stall potentially affecting redo logging, local checkpoints, and NDB Replication.

We fix this by closing the marked `CopyFrag` scan whenever a node failure occurs while the `CopyFrag` is waiting for a row lock. (Bug #34823988)

References: See also: Bug #35037327.

- In certain cases, invalid signal data was not handled correctly. (Bug #34787608)
- Following execution of `DROP NODEGROUP` in the management client, attempting to creating or altering an `NDB` table specifying an explicit number of partitions or using `MAX_ROWS` was rejected with `Got error 771 'Given NODEGROUP doesn't exist in this cluster' from NDB`. (Bug #34649576)
- In a cluster with multiple management nodes, when one management node connected and later disconnected, any remaining management nodes were not aware of this node and were eventually forced to shut down when stopped nodes reconnected; this happened whenever the cluster still had live data nodes.

  On investigation it was found that node disconnection handling was done in the `NF_COMPLETEREP` path in `ConfigManager` but the expected `NF_COMPLETEREP` signal never actually arrived. We solve this by handling disconnecting management nodes when the `NODE_FAILREP` signal arrives, rather than waiting for `NF_COMPLETEREP`. (Bug #34582919)
- When reorganizing a table with `ALTER TABLE ... REORGANIZE PARTITION` following addition of new data nodes to the cluster, unique hash indexes were not redistributed properly. (Bug #30049013)
- During a rolling restart of a cluster with two data nodes, one of them refused to start, reporting that the redo log fragment file size did not match the configured one and that an initial start of the node was required. Fixed by addressing a previously unhandled error returned by `fsync()`, and retrying the write. (Bug #28674694)
- For a partial local checkpoint, each fragment LCP must be to be able to determine the precise state of the fragment at the start of the LCP and the precise difference in the fragment between the start of the current LCP and the start of the previous one. This is tracked using row header information and page header information; in cases where physical pages are removed this is also tracked in logical page map information.

  A page included in the current LCP, before the LCP scan reaches it, is released due to the commit or rollback of some operation on the fragment, also releasing the last used storage on the page.

  Since the released page could not be found by the scan, the release itself set the `LCP_SCANNED_BIT` of the page map entry it was mapped into, in order to indicate that the page was already handled from the point of view of the current LCP, causing subsequent allocation and release of the pages mapped to the entry during the LCP to be ignored. The state of the entry at the start of the LCP was also set as allocated in the page map entry.

  These settings are cleared only when the next LCP is prepared. Any page release associated with the page map entry before the clearance would violate the requirement that the bit is not set; we resolve this issue by removing the (incorrect) requirement. (Bug #23539857)
- A data node could hit an overly strict assertion when the thread liveness watchdog triggered while the node was already shutting down. We fix the issue by relaxing this assertion in such cases. (Bug #22159697)
- Removed a leak of long message buffer memory that occurred each time an index was scanned for updating index statistics. (Bug #108043, Bug #34568135)

- Fixed an uninitialized variable in `Suma.cpp`. (Bug #106081, Bug #33764143)

Changes in MySQL NDB Cluster 7.6.24 (5.7.40-ndb-7.6.24) (2202-10-12)

Bugs Fixed

- `NdbScanOperation` errors are returned asynchronously to the client, possibly while the client is engaged in other processing. A successful call to `NdbTransaction::execute()` guarantees only that the scan request has been assembled and sent to the transaction coordinator without any errors; it does not wait for any sort of `CONF` or `REF` signal to be returned from the data nodes. In this particular case, the expected `TAB_SCANREF` signal was returned asynchronously into the client space, possibly while the client was still performing other operations.

  We make this behavior more deterministic by not setting the `NdbTransaction` error code when a `TAB_SCANREF` error is received. (Bug #34348706)

- The combination of batching with multiple in-flight operations per key, use of `IgnoreError`, and transient errors occurring on non-primary replicas led in some cases to inconsistencies within `DBTUP` resulting in replica misalignment and other issues. We now prevent this from happening by detecting when operations are failing on non-primary replicas, and forcing `AbortOnError` handling (rollback) in such cases for the containing transaction. (Bug #34013385)

- When the rate of changes was high, event subscribers were slow to acknowledge receipt, or both, it was possible for the `SUMA` block to run out of space for buffering events. (Bug #30467140)

- `ALTER TABLE ... COMMENT="NDB_TABLE=READ_BACKUP=1"` or `ALTER TABLE..COMMENT="NDB_TABLE=READ_BACKUP=0"` performs a non-copying (online) `ALTER` operation on a table to add or remove its `READ_BACKUP` property (see NDB_TABLE Options), which increments the index version of all indexes on the table. Existing statistics, stored using the previous index version, were orphaned and never deleted; this led to wasted memory and inefficient searches when collecting index statistics.

  We address these issues by cleaning up the index samples; we delete any samples whose sample version is greater than or less than the current sample version. In addition, when no existing statistics are found by index ID and version, and when indexes are dropped. In this last case, we relax the bounds for the delete operation and remove all entries corresponding to the index ID in question, as opposed to both index ID and index version.

  This fix cleans up the sample table which stores the bulk of index statistics data. The head table, which consists of index metadata rather than actual statistics, still contains orphaned rows, but since these occupy an insignificant amount of memory, they do not adversely affect statistics search efficiency, and stale entries are cleaned up when index IDs and versions are reused.

  See also NDB API Statistics Counters and Variables. (Bug #29611297)

Changes in MySQL NDB Cluster 7.6.23 (5.7.39-ndb-7.6.23) (2022-07-27)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change; NDB Replication:** The `slave_allow_batching` system variable affects how efficiently the slave applies epoch transactions. When this variable is set to `OFF`, by default, every discrete replication event in the binary log is applied and executed separately, which generally leads to poor performance.

Beginning with this release, if slave batching is disabled (`slave_allow_batching` set to `OFF`), the MySQL server now reports a warning, with an admonition to enable it.

Bugs Fixed

- **NDB Cluster APIs:** The internal function `NdbThread_SetThreadPrio()` sets the thread priority (`thread_prio`) for a given thread type when applying the setting of the `ThreadConfig` configuration parameter. It was possible for this function in some cases to return an error when it had actually succeeded, which could have a an unfavorable impact on the performance of some NDB API applications. (Bug #34038630)
- Path lengths were not always calculated correctly by the data nodes. (Bug #33993607)
- The internal function `NdbReceiver::unpackRecAttr()`, which unpacks attribute values from a buffer from a `GSN_TRANSID_AI` signal, did not check to ensure that attribute sizes fit within the buffer. This could corrupt the buffer which could in turn lead to reading beyond the buffer and copying beyond destination buffers. (Bug #33941167)
- Some `NDB` internal signals were not always checked properly. (Bug #33896428)

Changes in MySQL NDB Cluster 7.6.22 (5.7.38-ndb-7.6.22) (2022-04-27)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- The client receive thread was enabled only when under high load, where the criterion for determining “high load” was that the number of clients waiting in the poll queue (the receive queue) was greater than `min_active_clients_recv_thread` (default: `8`).

  This was a poor metric for determining high load, since a single client, such as the binary log injector thread handling incoming replication events, could experience high load on its own as well. The same was true of a pushed join query (in which very large batches of incoming `TRANSID_AI` signals are received).

  We change the receive thread such that it now sleeps in the poll queue rather than being deactivated completely, so that it is now always available for handling incoming signals, even when the client is not under high load. (Bug #33752914)

Bugs Fixed

- **Important Change:** The maximum value supported for the `--ndb-batch-size` server option has been increased from `31536000` to `2147483648` (2 GB). (Bug #21040523)
- **Performance:** When inserting a great many rows into an empty or small table in the same transaction, the rate at which rows were inserted quickly declined to less than 50% of the initial rate; subsequently, it was found that roughly 50% of all CPU time was spent in `Dbacc::getElement()`, and the root cause identified to be the timing of resizing the structures used for storing elements by `DBACC`, growing with the insertion of more rows in the same transaction, and shrinking following a commit.

  We fix this issue by checking for a need to resize immediately following the insertion or deletion of an element. This also handles the subsequent rejection of an insert. (Bug #33803487)

  References: See also: Bug #33803541.

- **Performance:** The internal function `computeXorChecksum()` was implemented such that great care was taken to aid the compiler in generating optimal code, but it was found that it consumed excessive CPU resources, and did not perform as well as a simpler implementation. This function is now reimplemented with a loop summing up `XOR` results over an array, which appears to result in better optimization with both GCC and Clang compilers. (Bug #33757412)

- 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)

- `NdbEventBuffer` hash key generation for non-character data reused the same 256 hash keys; in addition, strings of zero length were ignored when calculating hash keys. (Bug #33783274)

- The collection of NDB API statistics based on the `EventBytesRecvdCount` event counter incurred excessive overhead. Now this counter is updated using a value which is aggregated as the event buffer is filled, rather than traversing all of the event buffer data in a separate function call.

  For more information, see NDB API Statistics Counters and Variables. (Bug #33778923)

- The receiver thread ID was hard-coded in the internal method `TransporterFacade::raise_thread_prio()` such that it always acted to raise the priority of the receiver thread, even when called from the send thread. (Bug #33752983)

- The deprecated `-r` option for `ndbd` has been removed. In addition, this change also removes extraneous text from the output of `ndbd --help`. (Bug #33362935)

  References: See also: Bug #31565810.

- The `ndb_import` tool handled only the hidden primary key which is defined by `NDB` when a table does not have an explicit primary key. This caused an error when inserting a row containing `NULL` for an auto-increment primary key column, even though the same row was accepted by `LOAD DATA INFILE`.

  We fix this by adding support for importing a table with one or more instances of `NULL` in an auto-increment primary key column. This includes a check that a table has no more than one auto-increment column; if this column is nullable, it is redefined by `ndb_import` as `NOT NULL`, and any occurrence of `NULL` in this column is replaced by a generated auto-increment value before inserting the row into `NDB`. (Bug #30799495)

- When a node failure is detected, surviving nodes in the same nodegroup as this node attempt to resend any buffered change data to event subscribers. In cases in which there were no outstanding epoch deliveries, that is, the list of unacknowledged GCIs was empty, the surviving nodes made the incorrect assumption that this list would never be empty. (Bug #30509416)

Changes in MySQL NDB Cluster 7.6.21 (5.7.37-ndb-7.6.21) (2022-01-19)

Bugs Fixed

- **Important Change:** The deprecated data node option `--connect-delay` has been removed. This option was a synonym for `--connect-retry-delay`, which was not honored in all cases; this issue has been fixed, and the option now works correctly. In addition, the short form `-r` for this option has been deprecated, and you should expect it to be removed in a future release. (Bug #31565810)

  References: See also: Bug #33362935.

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

- Timing of `wait for scans` log output during online reorganization was not performed correctly. As part of this fix, we change timing to generate one message every 10 seconds rather than scaling indefinitely, so as to supply regular updates. (Bug #35523977)
- Added missing values checks in `ndbd` and `ndbmtd`. (Bug #33661024)
- Online table reorganization increases the number of fragments of a table, and moves rows between them. This is done in the following steps:

  1. Copy rows to new fragments
  2. Update distribution information (hashmap count and total fragments)
  3. Wait for scan activity using old distribution to stop
  4. Delete rows which have moved out of existing partitions
  5. Remove reference to old hashmap
  6. Wait for scan activity started since step 2 to stop

  Due to a counting error, it was possible for the reorganization to hang in step 6; the scan reference count was not decremented, and thus never reached zero as expected. (Bug #33523991)
- The same pushed join on `NDB` tables returned an incorrect result when the `batched_key_access` optimizer switch was enabled.

  This issue arose as follows: When the batch key access (BKA) algorithm is used to join two tables, a set of batched keys is first collected from one of the tables; a multirange read (MRR) operation is constructed against the other. A set of bounds (ranges) is specified on the MRR, using the batched keys to construct each bound.

  When result rows are returned it is necessary to identify which range each returned row comes from. This is used to identify the outer table row to perform the BKA join with. When the MRR operation in question was a root of a pushed join operation, `SPJ` was unable to retrieve this identifier (`RANGE_NO`). We fix this by implementing the missing `SPJ` API functionality for returning such a `RANGE_NO` from a pushed join query. (Bug #33416308)
- The MySQL Optimizer uses two different methods, `handler::read_cost()` and `Cost_model::page_read_cost()`, to estimate the cost for different access methods, but the cost values returned by these were not always comparable; in some cases this led to the wrong index being chosen and longer execution time for effected queries. To fix this for `NDB`, we override the optimizer's `page_read_cost()` method with one specific to `NDBCLUSTER`. It was also found while working on this issue that the `NDB` handler did not implement the `read_time()` method, used by `read_cost()`; this method is now implemented by `ha_ndbcluster`, and thus the optimizer can now properly take into account the cost difference for `NDB` when using a unique key as opposed to an ordered index (range scan). (Bug #33317872)
- In certain cases, an event's category was not properly detected. (Bug #33304814)
- `DBDICT` did not always perform table name checks correctly. (Bug #33161548)
- Added a number of missing ID and other values checks in `ndbd` and `ndbmtd`. (Bug #33161486, Bug #33162047)
- Added a number of missing ID and other values checks in `ndbd` and `ndbmtd`. (Bug #33161259, Bug #33161362)

- `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)
- The order of parameters used in the argument to `ndb_import --csvopt` is now handled consistently, with the rightmost parameter always taking precedence. This also applies to duplicate instances of a parameter. (Bug #32822757)
- In some cases, issues with the redo log while restoring a backup led to an unplanned shutdown of the data node. To fix this, when the redo log file is not available for writes, we now include the correct wait code and waiting log part in the `CONTINUEB` signal before sending it. (Bug #32733659)

  References: See also: Bug #31585833.
- When a local data manager had no fragments, the data node could not be restarted. While it is unusual, this lack of any fragments can occur when there have been changes in the number of LDMs since any tables were last created. (Bug #32288569)
- A query used by MySQL Enterprise Monitor to monitor memory use in NDB Cluster became markedly less performant as the number of `NDB` tables increased. We fix this as follows:
  - Row counts for virtual `ndbinfo` tables have been made available to the MySQL optimizer
  - Size estimates are now provided for all `ndbinfo` tables

  Following these improvements, queries against `ndbinfo` tables should be noticeably faster. (Bug #28658625)
- `NDB` did not close any pending schema transactions when returning an error from internal system table creation and drop functions.

Changes in MySQL NDB Cluster 7.6.20 (5.7.36-ndb-7.6.20) (2021-10-20)

Bugs Fixed

- 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)
- Added an `ndbrequire()` in `QMGR` to ensure that process reports from signal data use appropriate node IDs. (Bug #32983240)

- 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)

- Event buffer status messages shown by the event logger have been improved. Percentages are now displayed only when it makes to do so. In addition, if a maximum size is not defined, the printout shows `max=unlimited`. (Bug #21276857)

Changes in MySQL NDB Cluster 7.6.19 (5.7.35-ndb-7.6.19) (2021-07-21)

- Functionality Added or Changed

- Bugs Fixed

Functionality Added or Changed

- `ndb_restore` now supports conversion between `NULL` and `NOT NULL` columns, as follows:

  - To restore a `NULL` column as `NOT NULL`, use the `--lossy-conversions` option. The presence of any `NULL` rows in the column causes `ndb_restore` to raise an and exit.

  - To restore a `NOT NULL` column as `NULL`, use the `--promote-attributes` option.

  For more information, see the descriptions of the indicated `ndb_restore` options. (Bug #32702637)

Bugs Fixed

- **Packaging:** The `ndb-common` man page was removed, and the information it contained moved to other man pages. (Bug #32799519)

- **NDB Cluster APIs:** Added the `NDB_LE_EventBufferStatus3` log event type to `Ndb_logevent_type` in the MGM API. This is an extension of the `NDB_LE_EventBufferStatus` type which handles total, maximum, and allocated bytes as 64-bit values.

As part of this fix, the maximum value of the `ndb_eventbuffer_max_alloc` server system variable is increased to 9223372036854775807 (2^{63} - 1).

For more information, see The Ndb_logevent_type Type. (Bug #32381666)

- `Ndb_rep_tab_key` member variables were not null-terminated before being logged. (Bug #32841430)

  References: See also: Bug #32393245.

- Some error messages printed by `ndb_restore` tried to access transactions that were already closed for error information, resulting in an unplanned exit. (Bug #32815725)

- Returning an error while determining the number of partitions used by a `NDB` table caused the MySQL server to write `Incorrect information in table.frm file` to its error log, despite the fact that the indicated file did not exist. This also led to problems with flooding of the error log when users attempted to open `NDB` tables while the MySQL server was not actually connected to `NDB`.

  We fix this by changing the function that determines the number of partitions to return 0 whenever `NDB` is not available, thus deferring any error detection until the MySQL server is once again connected to `NDB`. (Bug #32713166)

- Using duplicate node IDs with `CREATE NODEGROUP` (for example, `CREATE NODEGROUP 11, 11`) could lead to an unplanned shutdown of the cluster. Now when this command includes duplicate node IDs, it raises an error. (Bug #32701583)

- Improved the performance of queries against the `ndbinfo.cluster_locks` table, which could in some cases run quite slowly. (Bug #32655988)

- It was possible in some cases to miss the end point of undo logging for a fragment. (Bug #32623528)

  References: See also: Bug #31774459.

- The `--resume` option for `ndb_import` did not work correctly unless the `--stats` option was also specified. (Bug #31107058)

- A `DELETE` statement whose `WHERE` clause referred to a `BLOB` column was not executed correctly. (Bug #13881465)

Changes in MySQL NDB Cluster 7.6.18 (5.7.34-ndb-7.6.18) (2021-04-21)

Bugs Fixed

- `ndb_mgmd` now exits gracefully in the event of a `SIGTERM` just as it does following a management client `SHUTDOWN` command. (Bug #32446105)

- A node was permitted during a restart to participate in a backup before it had completed recovery, instead of being made to wait until its recovery was finished. (Bug #32381165)

- Running out of disk space while performing an `NDB` backup could lead to an unplanned shutdown of the cluster. (Bug #32367250)

- The default number of partitions per node (shown in `ndb_desc` output as `PartitionCount`) is calculated using the lowest number of LDM threads employed by any single live node, and was done only once, even after data nodes left or joined the cluster, possibly with a new configuration changing the LDM thread count and thus the default partition count. Now in such cases, we make sure the default number of partitions per node is recalculated whenever data nodes join or leave the cluster. (Bug #32183985)

- The local checkpoint (LCP) mechanism was changed in NDB 7.6 such that it also detected idle fragments—that is, fragments which had not changed since the last LCP and thus required no on-disk metadata update. The LCP mechanism could then immediately proceed to handle the next fragment. When there were a great many such idle fragments, the CPU consumption required merely to loop through these became highly significant, causing latency spikes in user transactions.

  A 1 ms delay was already inserted between each such idle fragment being handled. Testing later showed this to be too short an interval, and that we are normally not in as great a hurry to complete these idle fragments as we previously believed.

  This fix extends the idle fragment delay time to 20 ms if there are no redo alerts indicating an urgent need to complete the LCP. In case of a low redo alert state we wait 5 ms instead, and for a higher alert state we fall back to the 1 ms delay. (Bug #32068551)

  References: See also: Bug #31655158, Bug #31613158.

- Event buffer congestion could lead to unplanned shutdown of SQL nodes which were performing binary logging. We fix this by updating the binary logging handler to use `Ndb::pollEvents2()` (rather than the deprecated `pollEvents()` method) to catch and handle such errors properly, instead. (Bug #31926584)

- Generation of internal statistics relating to `NDB` object counts was found to lead to an increase in transaction latency at very high rates of transactions per second, brought about by returning an excessive number of freed `NDB` objects. (Bug #31790329)

- Calculation of the redo alert state based on redo log usage was overly aggressive, and thus incorrect, when using more than 1 log part per LDM.

Changes in MySQL NDB Cluster 7.6.17 (5.7.33-ndb-7.6.17) (2021-01-19)

- 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

- While retrieving sorted results from a pushed-down join using `ORDER BY` with the `index` access method (and without `filesort`), an SQL node sometimes unexpectedly terminated. (Bug #32203548)

- Logging of redo log initialization showed log part indexes rather than log part numbers. (Bug #32200635)

- Signal data was overwritten (and lost) due to use of extended signal memory as temporary storage. Now in such cases, extended signal memory is not used in this fashion. (Bug #32195561)

- 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.

- As with writing redo log records, when the file currently used for writing global checkpoint records becomes full, writing switches to the next file. This switch is not supposed to occur until the new file is actually ready to receive the records, but no check was made to ensure that this was the case. This could lead to an unplanned data node shutdown restoring data from a backup using `ndb_restore`. (Bug #31585833)

- `ndb_restore` encountered intermittent errors while replaying backup logs which deleted blob values; this was due to deletion of blob parts when a main table row containing blob one or more values was deleted. This is fixed by modifying `ndb_restore` to use the asynchronous API for blob deletes, which does not trigger blob part deletes when a blob main table row is deleted (unlike the synchronous API), so that a delete log event for the main table deletes only the row from the main table. (Bug #31546136)

- 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 7.6.16 (5.7.32-ndb-7.6.16) (2020-10-20)

- 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)
- **NDB Cluster APIs:** In certain cases, the `Table::getColumn()` method returned the wrong `Column` object. This could happen when the full name of one table column was a prefix of the name of another, or when the names of two columns had the same hash value. (Bug #31774685)
- **NDB Cluster APIs:** It was possible to make invalid sequences of NDB API method calls using blobs. This was because some method calls implicitly cause transaction execution inline, to deal with blob parts and other issues, which could cause user-defined operations not to be handled correctly due to the use of a method executing operations relating to blobs while there still user-defined blob operations pending. Now in such cases, NDB raises a new error 4558 `Pending blob operations must be executed before this call`. (Bug #27772916)
- After encountering the data node in the configuration file which used `NodeGroup=65536`, the management server stopped assigning data nodes lacking an explicit `NodeGroup` setting to node groups. (Bug #31825181)
- In some cases, `QMGR` returned conflicting `NDB` engine and MySQL server version information, which could lead to unplanned management node shutdown. (Bug #31471959)
- During different phases of the restore process, `ndb_restore` used different numbers of retries for temporary errors as well as different sleep times between retries. This is fixed by implementing consistent retry counts and sleep times across all restore phases. (Bug #31372923)
- Backups errored out with **`FsErrInvalidParameters`** when the filesystem was running with `O_DIRECT` and a data file write was not aligned with the 512-byte block size used by `O_DIRECT` writes. If the total fragment size in the data file is not aligned with the `O_DIRECT` block size, `NDB` pads the last write to the required size, but when there were no fragments to write, `BACKUP` wrote only the header and footer to the data file. Since the header and footer are less than 512 bytes, leading to the issue with the `O_DIRECT` write.

  This is fixed by padding out the generic footer to 512 bytes if necessary, using an `EMPTY_ENTRY`, when closing the data file. (Bug #31180508)
- Altering the table comment of a fully replicated table using `ALGORITHM=INPLACE` led to an assertion. (Bug #31139313)
- Data nodes did not start when the `RealtimeScheduler` configuration parameter was set to 1. This was due to the fact that index builds during startup are performed by temporarily diverting some I/O threads for use as index building threads, and these threads inherited the realtime properties of the I/O threads. This caused a conflict (treated as a fatal error) when index build thread specifications were checked to ensure that they were not realtime threads. This is fixed by making sure that index build threads are not treated as realtime threads regardless of any settings applying to their host I/O threads, which is as actually intended in their design. (Bug #27533538)

Changes in MySQL NDB Cluster 7.6.15 (5.7.31-ndb-7.6.15) (2020-07-14)

Bugs Fixed

- **NDB Disk Data:** `ndbmtd` sometimes terminated unexpectedly when it could not complete a lookup for a log file group during a restore operation. (Bug #31284086)
- **NDB Disk Data:** An uninitialized variable led to issues when performing Disk Data DDL operations following a restart of the cluster. (Bug #30592528)

- `ndb_restore --remap-column` did not handle columns containing `NULL` values correctly. Now any offset specified by the mapping function used with this option is not applied to `NULL`, so that `NULL` is preserved as expected. (Bug #31966676)

- `MaxDiskWriteSpeedOwnRestart` was not honored as an upper bound for local checkpoint writes during a node restart. (Bug #31337487)

  References: See also: Bug #29943227.

- 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)

- `DUMP 1001` (`DumpPageMemoryOnFail`) now prints out information about the internal state of the data node page memory manager when allocation of pages fails due to resource constraints. (Bug #31231286)

- Statistics generated by `NDB` for use in tracking internal objects allocated and deciding when to release them were not calculated correctly, with the result that the threshold for resource usage was 50% higher than intended. This fix corrects the issue, and should allow for reduced memory usage. (Bug #31127237)

- 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)

- When responding to a `SCANTABREQ`, an API node can provide a distribution key if it knows that the scan should work on only one fragment, in which case the distribution key should be the fragment ID, but in some cases a hash of the partition key was used instead, leading to failures in `DBTC`. (Bug #30774226)

- Several memory leaks found in `ndb_import` have been removed. (Bug #30756434, Bug #30727956)

- The master node in a backup shut down unexpectedly on receiving duplicate replies to a `DEFINE_BACKUP_REQ` signal. These occurred when a data node other than the master errored out during the backup, and the backup master handled the situation by sending itself a `DEFINE_BACKUP_REF` signal on behalf of the missing node, which resulted in two replies being received from the same node (a `CONF` signal from the problem node prior to shutting down and the `REF` signal from the master on behalf of this node), even though the master expected only one reply per node. This scenario was also encountered for `START_BACKUP_REQ` and `STOP_BACKUP_REQ` signals.

  This is fixed in such cases by allowing duplicate replies when the error is the result of an unplanned node shutdown. (Bug #30589827)

- When updating `NDB_TABLE` comment options using `ALTER TABLE`, other options which has been set to non-default values when the table was created but which were not specified in the `ALTER TABLE` statement could be reset to their defaults.

  See Setting NDB Comment Options, for more information. (Bug #30428829)

- When, during a restart, a data node received a `GCP_SAVEREQ` signal prior to beginning start phase 9, and thus needed to perform a global checkpoint index write to a local data manager's local checkpoint control file, it did not record information from the `DIH` block originating with the node that sent the signal as part of the data written. This meant that, later in start phase 9, when attempting to send a `GCP_SAVECONF` signal in response to the `GCP_SAVEREQ`, this information was not available, which meant the response could not be sent, resulting in an unplanned shutdown of the data node. (Bug #30187949)

- Setting `EnableRedoControl` to `false` did not fully disable `MaxDiskWriteSpeed`, `MaxDiskWriteSpeedOtherNodeRestart`, and `MaxDiskWriteSpeedOwnRestart` as expected. (Bug #29943227)

  References: See also: Bug #31337487.

- Removed a memory leak found in the `ndb_import` utility. (Bug #29820879)

- A `BLOB` value is stored by `NDB` in multiple parts; when reading such a value, one read operation is executed per part. If a part is not found, the read fails with a `row not found error`, which indicates a corrupted `BLOB`, since a `BLOB` should never have any missing parts. A problem can arise because this error is reported as the overall result of the read operation, which means that mysqld sees no error and reports zero rows returned.

  This issue is fixed by adding a check specifically for the case in wich a blob part is not found. Now, when this occurs, overwriting the `row not found` error with `corrupted blob`, which causes the originating `SELECT` statement to fail as expected. Users of the NDB API should be aware that, despite this change, the `NdbBlob::getValue()` method continues to report the error as `row not found` in such cases. (Bug #28590428)

- Incorrect handling of operations on fragment replicas during node restarts could result in a forced shutdown, or in content diverging between fragment replicas, when primary keys with nonbinary (case-sensitive) equality conditions were used. (Bug #98526, Bug #30884622)

Changes in MySQL NDB Cluster 7.6.14 (5.7.30-ndb-7.6.14) (2020-04-28)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **NDB Client Programs:** Two options are added for the `ndb_blob_tool` utility, to enable it to detect missing blob parts for which inline parts exist, and to replace these with placeholder blob parts (consisting of space characters) of the correct length. To check whether there are missing blob parts, use the `ndb_blob_tool --check-missing` option. To replace with placeholders any blob parts which are missing, use the program's `--add-missing` option, also added in this release. (Bug #28583971)

- **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. (WL #13727, WL #13728)

- **MySQL NDB ClusterJ:** The unused `antlr3` plugin has been removed from the ClusterJ `pom` file. (Bug #29931625)

- **MySQL NDB ClusterJ:** The minimum Java version ClusterJ supports for MySQL NDB Cluster 8.0 is now Java 8. (Bug #29931625)

- **MySQL NDB ClusterJ:** A few Java APIs used by ClusterJ are now deprecated in recent Java versions. These adjustments have been made to ClusterJ:

  - Replaced all `Class.newInstance()` calls with `Class.getDeclaredConstructor().newInstance()` calls. Also updated the exception handling and the test cases wherever required.

  - All the `Number` classes' constructors that instantiate an object from a `String` or a primitive type are deprecated. Replaced all such deprecated instantiation calls with the corresponding `valueOf()` method calls.

  - The `Proxy.getProxyClass()` is now deprecated. The `DomainTypeHandlerImpl` class now directly creates a new instance using the `Proxy.newProxyInstance()` method; all references to the `Proxy` class and its constructors are removed from the `DomainTypeHandlerImpl` class. `SessionFactoryImpl` class now uses the interfaces underlying the proxy object to identify the domain class rather than using the Proxy class. Also updated `DomainTypeHandlerFactoryTest`.

  - The `finalize()` method is now deprecated. This patch does not change the overriding finalize() methods, but just suppresses the warnings on them. This deprecation will be handled separately in a later patch.

  - Updated the CMake configuration to treat deprecation warnings as errors when compiling ClusterJ.

  (Bug #29931625)

- It now possible to consolidate data from separate instances of NDB Cluster into a single target NDB Cluster when the original datasets all use the same schema. This is supported when using backups created using `START BACKUP` in `ndb_mgm` and restoring them with `ndb_restore`, using the `--remap-column` option implemented in this release (along with `--restore-data` and possibly other options). `--remap-column` can be employed to handle cases of overlapping primary, unique, or both sorts of

key values between source clusters, and you need to make sure that they do not overlap in the target cluster. This can also be done to preserve other relationships between tables.

When used together with `--restore-data`, the new option applies a function to the value of the indicated column. The value set for this option is a string of the format *`db.tbl.col:fn:args`*, whose components are listed here:

- *`db`*: Database name, after performing any renames.
- *`tbl`*: Table name.
- *`col`*: Name of the column to be updated. This column's type must be one of `INT` or `BIGINT`, and can optionally be `UNSIGNED`.
- *`fn`*: Function name; currently, the only supported name is `offset`.
- *`args`*: The size of the offset to be added to the column value by `offset`. The range of the argument is that of the signed variant of the column's type; thus, negative offsets are supported.

You can use `--remap-column` for updating multiple columns of the same table and different columns of different tables, as well as combinations of multiple tables and columns. Different offset values can be employed for different columns of the same table.

As part of this work, two new options are also added to `ndb_desc` in this release:

- `--auto-inc` (short form `-a`): Includes the next auto-increment value in the output, if the table has an `AUTO_INCREMENT` column.
- `--context` (short form `-x`): Provides extra information about the table, including the schema, database name, table name, and internal ID.

These options may be useful for obtaining information about `NDB` tables when planning a merge, particularly in situations where the `mysql` client may not be readily available.

For more information, see the descriptions for `--remap-column`, `--auto-inc`, and `--context`. (Bug #30383950, WL #11796)

- `ndb_restore` now supports different primary key definitions for source and target tables when restoring from an `NDB` native backup, using the `--allow-pk-changes` option introduced in this release. Both increasing and decreasing the number of columns making up the original primary key are supported. This may be useful when it is necessary to accommodate schema version changes while restoring data, or when doing so is more efficient or less time-consuming than performing `ALTER TABLE` statements involving primary key changes on a great many tables following the restore operation.

  When extending a primary key with additional columns, any columns added must not be nullable, and any values stored in any such columns must not change while the backup is being taken. Changes in the values of any such column while trying to add it to the table's primary key causes the restore operation to fail. Due to the fact that some applications set the values of all columns when updating a row even if the values of one or more of the columns does not change, it is possible to override this behavior by using

the `--ignore-extended-pk-updates` option which is also added in this release. If you do this, care must be taken to insure that such column values do not actually change.

When removing columns from the table's primary key, it is not necessary that the columns dropped from the primary key remain part of the table afterwards.

For more information, see the description of the `--allow-pk-changes` option in the documentation for `ndb_restore`. (Bug #26435136, Bug #30383947, Bug #30634010, WL #10730)

- 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

- **MySQL NDB ClusterJ:** When a `Date` value was read from a NDB cluster, ClusterJ sometimes extracted the wrong year value from the row. It was because the `Utility` class, when unpacking the `Date` value, wrongly extracted some extra bits for the year. This patch makes ClusterJ only extract the required bits. (Bug #30600320)

- **MySQL NDB ClusterJ:** When the cluster's `NdbOperation::AbortOption` type had the value of `AO_IgnoreOnError`, when there was a read error, ClusterJ took that as the row was missing and returned `null` instead of an exception. This was because with `AO_IgnoreOnErro`, the `execute()` method always returns a success code after each transaction, and ClusterJ is supposed to check for any errors in any of the individual operations; however, read operations were not checked by ClusterJ in the case. With this patch, read operations are now checked for errors after query executions, so that a reading error is reported as such. (Bug #30076276)

- When restoring signed auto-increment columns, `ndb_restore` incorrectly handled negative values when determining the maximum value included in the data. (Bug #30928710)

- 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 7.6.13 (5.7.29-ndb-7.6.13) (2020-01-14)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change:** It is now possible to divide a backup into slices and to restore these in parallel using two new options implemented for the `ndb_restore` utility, making it possible to employ multiple instances of `ndb_restore` to restore subsets of roughly the same size of the backup in parallel, which should help to reduce the length of time required to restore an NDB Cluster from backup.

The `--num-slices` options determines the number of slices into which the backup should be divided; `--slice-id` provides the ID of the slice (0 to 1 less than the number of slices) to be restored by `ndb_restore`.

Up to 1024 slices are supported.

For more information, see the descriptions of the `--num-slices` and `--slice-id` options. (Bug #30383937, WL #10691)

Bugs Fixed

- **Incompatible Change:** The minimum value for the `RedoOverCommitCounter` data node configuration parameter has been increased from 0 to 1. The minimum value for the `RedoOverCommitLimit` data node configuration parameter has also been increased from 0 to 1.

  You should check the cluster global configuration file and make any necessary adjustments to values set for these parameters before upgrading. (Bug #29752703)

- **Microsoft Windows; NDB Disk Data:** On Windows, restarting a data node other than the master when using Disk Data tables led to a failure in `TSMAN`. (Bug #97436, Bug #30484272)

- **NDB Disk Data:** Compatibility code for the Version 1 disk format used prior to the introduction of the Version 2 format in NDB 7.6 turned out not to be necessary, and is no longer used.

- A faulty `ndbrequire()` introduced when implementing partial local checkpoints assumed that `m_participatingLQH` must be clear when receiving `START_LCP_REQ`, which is not necessarily true when a failure happens for the master after sending `START_LCP_REQ` and before handling any `START_LCP_CONF` signals. (Bug #30523457)

- A local checkpoint sometimes hung when the master node failed while sending an `LCP_COMPLETE_REP` signal and it was sent to some nodes, but not all of them. (Bug #30520818)

- Added the `DUMP 9988` and `DUMP 9989` commands. (Bug #30520103)

- Execution of `ndb_restore --rebuild-indexes` together with the `--rewrite-database` and `--exclude-missing-tables` options did not create indexes for any tables in the target database. (Bug #30411122)

- When synchronizing extent pages it was possible for the current local checkpoint (LCP) to stall indefinitely if a `CONTINUEB` signal for handling the LCP was still outstanding when receiving the `FSWRITECONF` signal for the last page written in the extent synchronization page. The LCP could also be restarted if another page was written from the data pages. It was also possible that this issue caused `PREP_LCP` pages to be written at times when they should not have been. (Bug #30397083)

- 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.

- Data node failures with the error `Another node failed during system restart...` occurred during a partial restart. (Bug #30368622)

- If a `SYNC_EXTENT_PAGES_REQ` signal was received by `PGMAN` while dropping a log file group as part of a partial local checkpoint, and thus dropping the page locked by this block for processing next, the LCP terminated due to trying to access the page after it had already been dropped. (Bug #30305315)

- The wrong number of bytes was reported in the cluster log for a completed local checkpoint. (Bug #30274618)

  References: See also: Bug #29942998.

- The number of data bytes for the summary event written in the cluster log when a backup completed was truncated to 32 bits, so that there was a significant mismatch between the number of log records and the number of data records printed in the log for this event. (Bug #29942998)

- Using 2 LDM threads on a 2-node cluster with 10 threads per node could result in a partition imbalance, such that one of the LDM threads on each node was the primary for zero fragments. Trying to restore a multi-threaded backup from this cluster failed because the datafile for one LDM contained only the 12-byte data file header, which `ndb_restore` was unable to read. The same problem could occur in other cases, such as when taking a backup immediately after adding an empty node online.

  It was found that this occurred when `ODirect` was enabled for an EOF backup data file write whose size was less than 512 bytes and the backup was in the `STOPPING` state. This normally occurs only for an aborted backup, but could also happen for a successful backup for which an LDM had no fragments. We fix the issue by introducing an additional check to ensure that writes are skipped only if the backup actually contains an error which should cause it to abort. (Bug #29892660)

  References: See also: Bug #30371389.

- In some cases the `SignalSender` class, used as part of the implementation of `ndb_mgmd` and `ndbinfo`, buffered excessive numbers of unneeded `SUB_GCP_COMPLETE_REP` and `API_REGCONF` signals, leading to unnecessary consumption of memory. (Bug #29520353)

  References: See also: Bug #20075747, Bug #29474136.

- The setting for the `BackupLogBufferSize` configuration parameter was not honored. (Bug #29415012)

- 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.

- A transaction which inserts a child row may run concurrently with a transaction which deletes the parent row for that child. One of the transactions should be aborted in this case, lest an orphaned child row result.

  Before committing an insert on a child row, a read of the parent row is triggered to confirm that the parent exists. Similarly, before committing a delete on a parent row, a read or scan is performed to confirm that no child rows exist. When insert and delete transactions were run concurrently, their prepare and commit operations could interact in such a way that both transactions committed. This occurred because the triggered reads were performed using `LM_CommittedRead` locks (see `NdbOperation::LockMode`), which are not strong enough to prevent such error scenarios.

  This problem is fixed by using the stronger `LM_SimpleRead` lock mode for both triggered reads. The use of `LM_SimpleRead` rather than `LM_CommittedRead` locks ensures that at least one transaction aborts in every possible scenario involving transactions which concurrently insert into child rows and delete from parent rows. (Bug #22180583)

- 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 7.6.12 (5.7.28-ndb-7.6.12) (2019-10-15)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- `ndb_restore` now reports the specific `NDB` error number and message when it is unable to load a table descriptor from a backup `.ctl` file. This can happen when attempting to restore a backup taken from a later version of the NDB Cluster software to a cluster running an earlier version—for example, when the backup includes a table using a character set which is unknown to the version of `ndb_restore` being used to restore it. (Bug #30184265)

- The output from `DUMP 1000` in the `ndb_mgm` client has been extended to provide information regarding total data page usage. (Bug #29841454)

  References: See also: Bug #29929996.

Bugs Fixed

- **NDB Disk Data:** When a data node failed following creation and population of an `NDB` table having columns on disk, but prior to execution of a local checkpoint, it was possible to lose row data from the tablespace. (Bug #29506869)

- **MySQL NDB ClusterJ:** If ClusterJ was deployed as a separate module of a multi-module web application, when the application tried to create a new instance of a domain object, the exception `java.lang.IllegalArgumentException: non-public interface is not defined by the given loader` was thrown. It was because ClusterJ always tries to create a proxy class from which the domain object can be instantiated, and the proxy class is an implementation of the domain interface and the protected `DomainTypeHandlerImpl::Finalizable` interface. The class loaders of these two interfaces were different in the case, as they belonged to different modules running on the web server, so that when ClusterJ tried to create the proxy class using the domain object interface's class loader, the above-mentioned exception was thrown. This fix makes the `Finalization` interface public so that the class loader of the web application would be able to access it even if it belongs to a different module from that of the domain interface. (Bug #29895213)

- **MySQL NDB ClusterJ:** ClusterJ sometimes failed with a segmentation fault after reconnecting to an NDB Cluster. This was due to ClusterJ reusing old database metadata objects from the old connection. With the fix, those objects are discarded before a reconnection to the cluster. (Bug #29891983)

- Once a data node is started, 95% of its configured `DataMemory` should be available for normal data, with 5% to spare for use in critical situations. During the node startup process, all of its configured `DataMemory` is usable for data, in order to minimize the risk that restoring the node data fails due to running out of data memory due to some dynamic memory structure using more pages for the same data than when the node was stopped. For example, a hash table grows differently during a restart than it did previously, since the order of inserts to the table differs from the historical order.

  The issue raised in this bug report occurred when a check that the data memory used plus the spare data memory did not exceed the value set for `DataMemory` failed at the point where the spare memory was reserved. This happened as the state of the data node transitioned from starting to started, when reserving spare pages. After calculating the number of reserved pages to be used for spare memory, and then the number of shared pages (that is, pages from shared global memory) to be used for this, the number of reserved pages already allocated was not taken into consideration. (Bug #30205182)

  References: See also: Bug #29616383.

- When executing a global schema lock (GSL), NDB used a single Ndb_table_guard object for successive retires when attempting to obtain a table object reference; it was not possible for this to succeed after failing on the first attempt, since Ndb_table_guard assumes that the underlying object pointer is determined once only—at initialisation—with the previously retrieved pointer being returned from a cached reference thereafter.

  This resulted in infinite waits to obtain the GSL, causing the binlog injector thread to hang so that mysqld considered all NDB tables to be read-only. To avoid this problem, NDB now uses a fresh instance of Ndb_table_guard for each such retry. (Bug #30120858)

  References: This issue is a regression of: Bug #30086352.

- When starting, a data node's local sysfile was not updated between the first completed local checkpoint and start phase 50. (Bug #30086352)

- In the BACKUP block, the assumption was made that the first record in c_backups was the local checkpoint record, which is not always the case. Now NDB loops through the records in c_backups to find the (correct) LCP record instead. (Bug #30080194)

- During node takeover for the master it was possible to end in the state LCP_STATUS_IDLE while the remaining data nodes were reporting their state as LCP_TAB_SAVED. This led to failure of the node when attempting to handle reception of a LCP_COMPLETE_REP signal since this is not expected when idle. Now in such cases local checkpoint handling is done in a manner that ensures that this node finishes in the proper state (LCP_TAB_SAVED). (Bug #30032863)

- Restoring tables for which MAX_ROWS was used to alter partitioning from a backup made from NDB 7.4 to a cluster running NDB 7.6 did not work correctly. This is fixed by ensuring that the upgrade code handling PartitionBalance supplies a valid table specification to the NDB dictionary. (Bug #29955656)

- During upgrade of an NDB Cluster when half of the data nodes were running NDB 7.6 while the remainder were running NDB 8.0, attempting to shut down those nodes which were running NDB 7.6 led to failure of one node with the error CHECK FAILEDNODEPTR.P->DBLQHFAI. (Bug #29912988, Bug #30141203)

- When performing a local checkpoint (LCP), a table's schema version was intermittently read as 0, which caused NDB LCP handling to treat the table as though it were being dropped. This could effect rebuilding of indexes offline by ndb_restore while the table was in the TABLE_READ_ONLY state. Now the function reading the schema version (getCreateSchemaVersion()) no longer not changes it while the table is read-only. (Bug #29910397)

- NDB index statistics are calculated based on the topology of one fragment of an ordered index; the fragment chosen in any particular index is decided at index creation time, both when the index is originally created, and when a node or system restart has recreated the index locally. This calculation is based in part on the number of fragments in the index, which can change when a table is reorganized. This means that, the next time that the node is restarted, this node may choose a different fragment, so that no fragments, one fragment, or two fragments are used to generate index statistics, resulting in errors from ANALYZE TABLE.

  This issue is solved by modifying the online table reorganization to recalculate the chosen fragment immediately, so that all nodes are aligned before and after any subsequent restart. (Bug #29534647)

- 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)

- The process of selecting the transaction coordinator checked for “live” data nodes but not necessarily for those that were actually available. (Bug #27160203)

Changes in MySQL NDB Cluster 7.6.11 (5.7.27-ndb-7.6.11) (2019-07-23)

- 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. (WL #12303)

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. (WL #13117)
- A pushed join with `ORDER BY` did not always return the rows of the result in the specified order. This could occur when the optimizer used an ordered index to provide the ordering and the index used a column from the table that served as the root of the pushed join. (Bug #29860378)
- 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)
- Lack of `SharedGlobalMemory` was incorrectly reported as lack of undo buffer memory, even though the cluster used no disk data tables. (Bug #29806771)

  References: This issue is a regression of: Bug #92125, Bug #28537319.
- Long `TCKEYREQ` signals did not always use the expected format when invoked from `TCINDXREQ` processing. (Bug #29772731)
- Improved error message printed when the maximum offset for a `FIXED` column is exceeded. (Bug #29714670)
- Data nodes could fail due to an assert in the `DBTC` block under certain circumstances in resource-constrained environments. (Bug #29528188)
- An upgrade to NDB 7.6.9 or later from an earlier version could not be completed successfully if the redo log was filled to more than 25% of capacity. (Bug #29506844)

- When the DBSPJ block called the internal function lookup_resume() to schedule a previously enqueued operation, it used a correlation ID which could have been produced from its immediate ancestor in the execution order, and not its parent in the query tree as assumed. This could happen during execution of a SELECT STRAIGHT_JOIN query.

  Now NDB checks whether the execution ancestor is different from the query tree parent, and if not, performs a lookup of the query tree parent, and the parent's correlation ID is enqueued to be executed later. (Bug #29501263)

- When a new master took over, sending a MASTER_LCP_REQ signal and executing MASTER_LCPCONF from participating nodes, it expected that they had not completed the current local checkpoint under the previous master, which need not be true. (Bug #29487340, Bug #29601546)

- 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 7.6.10 (5.7.26-ndb-7.6.10) (2019-04-26)

Bugs Fixed

- **NDB Disk Data:** The error message returned when validation of MaxNoOfOpenFiles in relation to InitialNoOfOpenFiles failed has been improved to make the nature of the problem clearer to users. (Bug #28943749)

- **NDB Disk Data:** Repeated execution of ALTER TABLESPACE ... ADD DATAFILE against the same tablespace caused data nodes to hang and left them, after being killed manually, unable to restart. (Bug #22605467)

- **NDB Cluster APIs:** NDB now identifies short-lived transactions not needing the reduction of lock contention provided by NdbBlob::close() and no longer invokes this method in cases (such as when autocommit is enabled) in which unlocking merely causes extra work and round trips to be performed prior to committing or aborting the transaction. (Bug #29305592)

  References: See also: Bug #49190, Bug #11757181.

- **NDB Cluster APIs:** When the most recently failed operation was released, the pointer to it held by NdbTransaction became invalid and when accessed led to failure of the NDB API application. (Bug #29275244)

- 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.

- API and data nodes running NDB 7.6 and later could not use an existing parsed configuration from an earlier release series due to being overly strict with regard to having values defined for configuration parameters new to the later release, which placed a restriction on possible upgrade paths. Now NDB 7.6 and later are less strict about having all new parameters specified explicitly in the configuration which they are served, and use hard-coded default values in such cases. (Bug #28993400)

- Added DUMP 406 (NdbfsDumpRequests) to provide NDB file system information to global checkpoint and local checkpoint stall reports in the node logs. (Bug #28922609)

- A race condition between the DBACC and DBLQH kernel blocks occurred when different operations in a transaction on the same row were concurrently being prepared and aborted. This could result in DBTUP attempting to prepare an operation when a preceding operation had been aborted, which was unexpected and could thus lead to undefined behavior including potential data node failures. To solve this issue, DBACC and DBLQH now check that all dependencies are still valid before attempting to prepare an operation.

  Note

  This fix also supersedes a previous one made for a related issue which was originally reported as Bug #28500861.

  (Bug #28893633)

- The ndbinfo.cpustat table reported inaccurate information regarding send threads. (Bug #28884157)

- Execution of an LCP_COMPLETE_REP signal from the master while the LCP status was IDLE led to an assertion. (Bug #28871889)

- Issuing a STOP command in the ndb_mgm client caused ndbmtd processes which had recently been added to the cluster to hang in Phase 4 during shutdown. (Bug #28772867)

- 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)

- After a commit failed due to an error, `mysqld` shut down unexpectedly while trying to get the name of the table involved. This was due to an issue in the internal function `ndbcluster_print_error()`. (Bug #28435082)

- `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.

- Neither the `MAX_EXECUTION_TIME` optimizer hint nor the `max_execution_time` system variable was respected for DDL statements or queries against `INFORMATION_SCHEMA` tables while an `NDB` global schema lock was in effect. (Bug #27538139)

- When query memory was exhausted in the `DBSPJ` kernel block while storing correlation IDs for deferred operations, the query was aborted with error status 20000 `Query aborted due to out of query memory`. (Bug #26995027)

  References: See also: Bug #86537.

- `MaxBufferedEpochs` is used on data nodes to avoid excessive buffering of row changes due to lagging `NDB` event API subscribers; when epoch acknowledgements from one or more subscribers lag by this number of epochs, an asynchronous disconnection is triggered, allowing the data node to release the buffer space used for subscriptions. Since this disconnection is asynchronous, it may be the case that it has not completed before additional new epochs are completed on the data node, resulting in new epochs not being able to seize GCP completion records, generating warnings such as those shown here:

  ```
  [ndbd] ERROR    -- c_gcp_list.seize() failed...

  ...

  [ndbd] WARNING  -- ACK wo/ gcp record...
  ```

  And leading to the following warning:

  ```
  Disconnecting node %u because it has exceeded MaxBufferedEpochs
  (100 > 100), epoch ....
  ```

  This fix performs the following modifications:

  - Modifies the size of the GCP completion record pool to ensure that there is always some extra headroom to account for the asynchronous nature of the disconnect processing previously described, thus avoiding `c_gcp_list` seize failures.
  - Modifies the wording of the `MaxBufferedEpochs` warning to avoid the contradictory phrase “100 > 100”.

  (Bug #20344149)

- 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 7.6.9 (5.7.25-ndb-7.6.9) (2019-01-22)

Bugs Fixed

- **Important Change:** When restoring to a cluster using data node IDs different from those in the original cluster, ndb_restore tried to open files corresponding to node ID 0. To keep this from happening, the --nodeid and --backupid options—neither of which has a default value—are both now explicitly required when invoking ndb_restore. (Bug #28813708)

- **Packaging; MySQL NDB ClusterJ:** libndbclient was missing from builds on some platforms. (Bug #28997603)

- **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 Cluster APIs:** When the NDB kernel's SUMA block sends a TE_ALTER event, it does not keep track of when all fragments of the event are sent. When NDB receives the event, it buffers the fragments, and processes the event when all fragments have arrived. An issue could possibly arise for very large table definitions, when the time between transmission and reception could span multiple epochs; during this time, SUMA could send a SUB_GCP_COMPLETE_REP signal to indicate that it has sent all data for an epoch, even though in this case that is not entirely true since there may be fragments of a TE_ALTER event still waiting on the data node to be sent. Reception of the SUB_GCP_COMPLETE_REP leads to closing the buffers for that epoch. Thus, when TE_ALTER finally arrives, NDB assumes that it is a duplicate from an earlier epoch, and silently discards it.

  We fix the problem by making sure that the SUMA kernel block never sends a SUB_GCP_COMPLETE_REP for any epoch in which there are unsent fragments for a SUB_TABLE_DATA signal.

  This issue could have an impact on NDB API applications making use of TE_ALTER events. (SQL nodes do not make any use of TE_ALTER events and so they and applications using them were not affected.) (Bug #28836474)

- Where a data node was restarted after a configuration change whose result was a decrease in the sum of MaxNoOfTables, MaxNoOfOrderedIndexes, and MaxNoOfUniqueHashIndexes, it sometimes failed with a misleading error message which suggested both a temporary error and a bug, neither of which was the case.

  The failure itself is expected, being due to the fact that there is at least one table object with an ID greater than the (new) sum of the parameters just mentioned, and that this table cannot be restored since the maximum value for the ID allowed is limited by that sum. The error message has been changed to reflect this, and now indicates that this is a permanent error due to a problem configuration. (Bug #28884880)

- 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.

- Running `ANALYZE TABLE` on an `NDB` table with an index having longer than the supported maximum length caused data nodes to fail. (Bug #28714864)
- It was possible in certain cases for nodes to hang during an initial restart. (Bug #28698831)

  References: See also: Bug #27622643.
- The output of `ndb_config --configinfo --xml --query-all` now shows that configuration changes for the `ThreadConfig` and `MaxNoOfExecutionThreads` data node parameters require system initial restarts (`restart="system" initial="true"`). (Bug #28494286)
- API nodes should observe that a node is moving through `SL_STOPPING` phases (graceful stop) and stop using the node for new transactions, which minimizes potential disruption in the later phases of the node shutdown process. API nodes were only informed of node state changes via periodic heartbeat signals, and so might not be able to avoid interacting with the node shutting down. This generated unnecessary failures when the heartbeat interval was long. Now when a data node is being gracefully stopped, all API nodes are notified directly, allowing them to experience minimal disruption. (Bug #28380808)
- Executing `SELECT * FROM INFORMATION_SCHEMA.TABLES` caused SQL nodes to restart in some cases. (Bug #27613173)
- When scanning a row using a `TUP` scan or `ACC` scan, or when performing a read using the primary key, it is possible to start a read of the row and hit a real-time break during which it is necessary to wait for the page to become available in memory. When the page request returns later, an attempt to read the row fails due to an invalid checksum; this is because, when the row is deleted, its checksum is invalidated.

  This problem is solved by introducing a new tuple header `DELETE_WAIT` flag, which is checked before starting any row scan or PK read operations on the row where disk data pages are not yet available, and cleared when the row is finally committed. (Bug #27584165, Bug #93035, Bug #28868412)
- 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)
- `mysqld` shut down unexpectedly when a purge of the binary log was requested before the server had completely started, and it was thus not yet ready to delete rows from the `ndb_binlog_index` table. Now when this occurs, requests for any needed purges of the `ndb_binlog_index` table are saved in a queue and held for execution when the server has completely started. (Bug #25817834)

- 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)

- `NdbScanFilter` did not always handle `NULL` according to the SQL standard, which could result in sending non-qualifying rows to be filtered (otherwise not necessary) by the MySQL server. (Bug #92407, Bug #28643463)

  References: See also: Bug #93977, Bug #29231709.

- `NDB` attempted to use condition pushdown on greater-than (`>`) and less-than (`<`) comparisons with `ENUM` column values but this could cause rows to be omitted in the result. Now such comparisons are no longer pushed down. Comparisons for equality (`=`) and inequality (`<>` / `!=`) with `ENUM` values are not affected by this change, and conditions including these comparisons can still be pushed down. (Bug #92321, Bug #28610217)

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

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Performance:** This release introduces a number of significant improvements in the performance of scans; these are listed here:

  - Row checksums help detect hardware issues, but do so at the expense of performance. `NDB` now offers the possibility of disabling these by setting the new `ndb_row_checksum` server system variable to 0; doing this means that row checksums are not used for new or altered tables. This can have a significant impact (5 to 10 percent, in some cases) on performance for all types of queries. This variable is set to 1 by default, to provide compatibility with the previous behavior.

  - A query consisting of a scan can execute for a longer time in the LDM threads when the queue is not busy.

  - Previously, columns were read before checking a pushed condition; now checking of a pushed condition is done before reading any columns.

  - Performance of pushed joins should see significant improvement when using range scans as part of join execution.

  (WL #11722)

Bugs Fixed

- **Packaging:** Expected NDB header files were in the `devel` RPM package instead of `libndbclient-devel`. (Bug #84580, Bug #26448330)
- **NDB Disk Data:** While restoring a local checkpoint, it is possible to insert a row that already exists in the database; this is expected behavior which is handled by deleting the existing row first, then inserting the new copy of that row. In some cases involving data on disk, `NDB` failed to delete the existing row. (Bug #91627, Bug #28341843)
- **NDB Client Programs:** Removed a memory leak in `NdbImportUtil::RangeList` that was revealed in ASAN builds. (Bug #91479, Bug #28264144)
- **MySQL NDB ClusterJ:** When a table containing a `BLOB` or a `TEXT` field was being queried with ClusterJ for a record that did not exist, an exception ("`The method is not valid in current blob state`") was thrown. (Bug #28536926)
- **MySQL NDB ClusterJ:** A `NullPointerException` was thrown when a full table scan was performed with ClusterJ on tables containing either a BLOB or a TEXT field. It was because the proper object initializations were omitted, and they have now been added by this fix. (Bug #28199372, Bug #91242)
- When copying deleted rows from a live node to a node just starting, it is possible for one or more of these rows to have a global checkpoint index equal to zero. If this happened at the same time that a full local checkpoint was started due to the undo log getting full, the `LCP_SKIP` bit was set for a row having GCI = 0, leading to an unplanned shutdown of the data node. (Bug #28372628)
- `ndbmtd` sometimes experienced a hang when exiting due to log thread shutdown. (Bug #28027150)
- When the `SUMA` kernel block receives a `SUB_STOP_REQ` signal, it executes the signal then replies with `SUB_STOP_CONF`. (After this response is relayed back to the API, the API is open to send more `SUB_STOP_REQ` signals.) After sending the `SUB_STOP_CONF`, SUMA drops the subscription if no subscribers are present, which involves sending multiple `DROP_TRIG_IMPL_REQ` messages to `DBTUP`. LocalProxy can handle up to 21 of these requests in parallel; any more than this are queued in the Short Time Queue. When execution of a `DROP_TRIG_IMPL_REQ` was delayed, there was a chance for the queue to become overloaded, leading to a data node shutdown with `Error in short time queue`.

  This issue is fixed by delaying the execution of the `SUB_STOP_REQ` signal if `DBTUP` is already handling `DROP_TRIG_IMPL_REQ` signals at full capacity, rather than queueing up the `DROP_TRIG_IMPL_REQ` signals. (Bug #26574003)
- Having a large number of deferred triggers could sometimes lead to job buffer exhaustion. This could occur due to the fact that a single trigger can execute many operations—for example, a foreign key parent trigger may perform operations on multiple matching child table rows—and that a row operation on a base table can execute multiple triggers. In such cases, row operations are executed in batches. When execution of many triggers was deferred—meaning that all deferred triggers are executed at pre-commit—the resulting concurrent execution of a great many trigger operations could cause the data node job buffer or send buffer to be exhausted, leading to failure of the node.

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

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

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

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

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

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

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

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

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

  References: See also: Bug #28893633.

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

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

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

- In some cases, a workload that included a high number of concurrent inserts caused data node failures when using debug builds. (Bug #91764, Bug #28387450, Bug #29055038)

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

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

- In certain cases, a cascade update trigger was fired repeatedly on the same record, which eventually consumed all available concurrent operations, leading to Error 233 `Out of operation records in transaction coordinator (increase MaxNoOfConcurrentOperations)`. If `MaxNoOfConcurrentOperations` was set to a value sufficiently high to avoid this, the issue

manifested as data nodes consuming very large amounts of CPU, very likely eventually leading to a timeout. (Bug #91472, Bug #28262259)

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

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

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

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

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

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

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

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

  - The default value of `RecoveryWork` is increased from 50 to 60 (60% of storage reserved for LCP files).

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

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

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

  See What is New in NDB Cluster 7.6, as well as the descriptions of the parameters mentioned previously, for more information. (Bug #90709, Bug #27942974, Bug #27942583, WL #9638)

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

Bugs Fixed

- **ndbinfo Information Database:** It was possible following a restart for (sometimes incomplete) fallback data to be used in populating the `ndbinfo.processes` table, which could lead to rows in this table with empty `process_name` values. Such fallback data is no longer used for this purpose. (Bug #27985339)
- **NDB Client Programs:** The executable file `host_info` is no longer used by `ndb_setup.py`. This file, along with its parent directory `share/mcc/host_info`, has been removed from the NDB Cluster distribution.

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

  References: See also: Bug #27621546.
- **MySQL NDB ClusterJ:** ClusterJ could not be built from source using JDK 9. (Bug #27977985)
- An `NDB` restore operation failed under the following conditions:
  - A data node was restarted
  - The local checkpoint for the fragment being restored used two `.ctl` files
  - The first of these `.ctl` files was the file in use
  - The LCP in question consisted of more than 2002 parts

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

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

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

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

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

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

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

- The previously experimental shared memory (SHM) transporter is now supported in production. SHM works by transporting signals through writing them into memory, rather than on a socket. NDB already attempts to use SHM automatically between a data node and an API node sharing the same host. To enable explicit shared memory connections, set the `UseShm` configuration parameter to `1` for the relevant data node. When explicitly defining shared memory as the connection method, it is also necessary that the data node is identified by `HostName` and the API node by `HostName`.

  Additional tuning parameters such as `ShmSize`, `ShmSpintime`, and `SendBufferMemory` can be employed to improve performance of the SHM transporter. Configuration of SHM is otherwise similar to that of the TCP transporter. The `SigNum` parameter is no longer used, and any settings made for it are now ignored. NDB Cluster Shared Memory Connections, provides more information about these parameters.

  In addition, as part of this work, `NDB` code relating to support for the legacy SCI transporter, which had long been unsupported, has been removed. See `www.dolphinics.com` for information about support for legacy SCI hardware or information about the newer Dolphin Express hardware. (WL #7512)

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

  Note

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

  (WL #11164)

- The poll receiver which `NDB` uses to read from sockets, execute messages from the sockets, and wake up other threads now offloads wakeup of other threads to a new thread that wakes up the other threads on request, and otherwise simply sleeps. This improves the scalability of a single cluster connection by keeping the receive thread from becoming overburdened by tasks including wakeup of other threads. (WL #9663)

Bugs Fixed

- **Important Change; NDB Client Programs:** `ndb_top` ignored short forms of command-line options, and did not in all cases handle misformed long options correctly. As part of the fix for these issues, the following changes have been made to command-line options used with `ndb_top` to bring them more into line with those used with other NDB Cluster and MySQL programs:

  - The `--passwd` option is removed, and replaced by `--password` (short form `-p`).

  - The short form `-t` for the `--port` option has been replaced by `-P`.

  - The short form `-x` for the `--text` option has been replaced by `-t`.

(Bug #26907833)

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

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

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

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

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

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

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

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

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

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

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

- An `NDB` online backup consists of data, which is fuzzy, and a redo and undo log. To restore to a consistent state it is necessary to ensure that the log contains all of the changes spanning the capture of the fuzzy data portion and beyond to a consistent snapshot point. This is achieved by waiting for a GCI

boundary to be passed after the capture of data is complete, but before stopping change logging and recording the stop GCI in the backup's metadata.

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

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

References: See also: Bug #27566346.

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

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

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

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

  References: See also: Bug #28728603.

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

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

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

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

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

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

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

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

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

  References: See also: Bug #20112700.

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

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

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

Bugs Fixed

- **NDB Client Programs:** On Unix platforms, the Auto-Installer failed to stop the cluster when `ndb_mgmd` was installed in a directory other than the default. (Bug #89624, Bug #27531186)
- **NDB Client Programs:** The Auto-Installer did not provide a mechanism for setting the `ServerPort` parameter. (Bug #89623, Bug #27539823)
- An internal buffer being reused immediately after it had been freed could lead to an unplanned data node shutdown. (Bug #27622643)

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

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

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

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

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

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

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

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

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

  Important

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

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

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

  In addition, the default value for `StartPartitionedTimeout` is changed from 60000 to 0.

This release also deprecates the data node configuration parameters `BackupDataBufferSize`, `BackupWriteSize`, and `BackupMaxWriteSize`; these are now subject to removal in a future NDB Cluster version. (Bug #27308632, WL #8069, WL #10302, WL #10993)

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

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

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

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

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

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

  See The NDB Cluster Auto-Installer (NDB 7.5) (NO LONGER SUPPORTED), for more information. (WL #10340, WL #10408, WL #10449)

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

  - Added a mechanism for setting SSH membership on a per-host basis.
  - Updated the Paramiko Python module to the most recent available version (2.6.1).
  - Provided a place in the GUI for encrypted private key passwords, and discontinued use of hardcoded passwords.

  Related enhancements implemented in the current release include the following:

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

  See The NDB Cluster Auto-Installer (NDB 7.5) (NO LONGER SUPPORTED), for more information. (WL #10426, WL #11128, WL #11289)

- **MySQL NDB ClusterJ:** ClusterJ now supports CPU binding for receive threads through the setRecvThreadCPUids() and getRecvThreadCPUids() methods. Also, the receive thread activation threshold can be set and get with the setRecvThreadActivationThreshold() and getRecvThreadActivationThreshold() methods. (WL #10815)

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

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

  This enhancement is implemented as follows:

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

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

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

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

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

  Note

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

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

  (Bug #25428560)

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

```
mysql> SELECT * FROM ndbinfo.error_messages WHERE error_code='321';
+------------+----------------------+-----------------+----------------------+
| error_code | error_description    | error_status    | error_classification |
+------------+----------------------+-----------------+----------------------+
```

```
|        321 | Invalid nodegroup id | Permanent error | Application error    |
+------------+----------------------+-----------------+----------------------+
1 row in set (0.00 sec)
```

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

- `ThreadConfig` now has an additional `nosend` parameter that can be used to prevent a `main`, `ldm`, `rep`, or `tc` thread from assisting the send threads, by setting this parameter to 1 for the given thread. By default, `nosend` is 0. It cannot be used with threads other than those of the types just listed. (WL #11554)

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

  Since the effect of this change is generally to require fewer SPJ requests and instances, performance of pushed-down joins should be improved in many cases. (WL #10234)

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

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

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

  Increasing the appropriate batch size or sizes can help amortize inter-thread and inter-node latencies and make use of more parallel resources (local and remote) to help scale DDL performance. (WL #11158)

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

  There are no user-visible changes in functionality directly associated with this work; this performance enhancement is part of the work being done in NDB 7.6 to improve undo long handling for partial local checkpoints. (WL #8478)

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

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

  Mapping from page ID to table ID and fragment ID is now done using information the extent header contains about the table IDs and fragment IDs of the pages used in a given extent. Since the extent pages are always present in the page cache, no extra disk reads are required to perform the mapping, and the information can be read using existing `TSMAN` data structures. (WL #10194)

- Added the `NODELOG DEBUG` command in the `ndb_mgm` client to provide runtime control over data node debug logging. `NODE DEBUG ON` causes a data node to write extra debugging information to its node log, the same as if the node had been started with `--verbose`. `NODELOG DEBUG OFF` disables the extra logging. (WL #11216)

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

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

  This parameter takes an integer value between 0 and 16, with 0 being the default; using 0 is the same as leaving `LocationDomainId` unset. (WL #10172)

Bugs Fixed

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  - A new error code is added to assist testing.

  (Bug #26364729)

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

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

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

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

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

  References: See also: Bug #27067538.

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

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

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

- When representing a materialized 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)

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

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

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

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

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

  References: See also: Bug #20981491.

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

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

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

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

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

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

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

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

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

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

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

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

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

- Functionality Added or Changed
- Bugs Fixed

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)
- **MySQL NDB ClusterJ:** A new automatic reconnection feature has been implemented to facilitate the handling of connectivity issues. The feature is enabled by setting a positive number for a new connection property, `com.mysql.clusterj.connection.autoreconnect.timeout`, which specifies the length of the timeout period in seconds. If a connectivity error occurs, ClusterJ attempts to reconnect the application to the NDB Cluster after the application closes all the sessions; if the application does not close all sessions within the timeout period, ClusterJ closes any open sections forcibly, and then attempts reconnection. See Error Handling and Reconnection for details. (WL #9545)
- In some critical situations such as data node failure, it was possible for the volume of log messages produced to cause file system and other issues, which compounded the problem, due to the fact that these messages were logged synchronously using `stdout`. To keep this from happening, log messages from worker threads now use a log buffer instead, which is nonblocking, and thus much less likely to cause interference with other processes under such conditions. (Bug #24748843)
- Added the `--diff-default` option for `ndb_config`. This option causes the program to print only those parameters having values that differ from their defaults. (Bug #85831, Bug #25844166)
- Added the `ndb_top` program on unix-based platforms. This utility shows CPU load and usage information for an `NDB` data node, with periodic updates (each second, by default). The display is in text or color ASCII graph format; both formats can be displayed at the same time. It is also possible to disable color output for the graph.

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

  `ndb_top` is not currently available for Windows platforms.

  For more information, see ndb_top — View CPU usage information for NDB threads. (WL #9788)

Bugs Fixed

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

  (Bug #85679, Bug #25799465)
- **NDB Disk Data:** If the tablespace for a disk table had been fully consumed when a node failed, and table rows were deleted and inserted—or updated with shrinking or expanding disk column values—while the node was unavailable, a subsequent restart could fail with error 1601 `Out of extents, tablespace full`. We prevent this from happening by reserving 4 percent of the tablespace for use during node starts. (Bug #25923125)
- **NDB Cluster APIs:** The implementation method `NdbDictionary::NdbTableImpl::getColumn()`, used from many places in the NDB API where a column is referenced by name, has been made more efficient. This method used a linear search of an array of columns to find the correct column object,

which could be inefficient for tables with many columns, and was detected as a significant use of CPU in customer applications. (Ideally, users should perform name-to-column object mapping, and then use column IDs or objects in method calls, but in practice this is not always done.) A less costly hash index implementation, used previously for the name lookup, is reinstated for tables having relatively many columns. (A linear search continues to be used for tables having fewer columns, where the difference in performance is neglible.) (Bug #24829435)

- **NDB Cluster APIs:** NDB error 631 is reclassified as the (temporary) node recovery error `Scan take over error, restart scan transaction`. This was previously exposed to applications as an internal (and permanent) error which provided no description. (Bug #86401, Bug #26116231)
- **MySQL NDB ClusterJ:** The JTie and NDB JTie tests were skipped when the unit tests for ClusterJ were being run. (Bug #26088583)
- **MySQL NDB ClusterJ:** Compilation for the tests for NDB JTie failed. It was due to how null references were handled, which has been corrected by this fix. (Bug #26080804)
- Backup `.log` files contained log entries for one or more extra fragments, due to an issue with filtering out changes logged by other nodes in the same node group. This resulted in a larger `.log` file and thus use of more resources than necessary; it could also cause problems when restoring, since backups from different nodes could interfere with one another while the log was being applied. (Bug #25891014)
- Memory exhaustion during fragment creation led to an unplanned shutdown of the cluster. This issue could be triggered by the addition of unique keys to a large number of columns at the same time. (Bug #25851801)
- When making the final write to a redo log file, it is expected that the next log file is already opened for writes, but this was not always the case with a slow disk, leading to node failure. Now in such cases `NDB` waits for the next file to be opened properly before attempting to write to it. (Bug #25806659)
- Data node threads can be bound to a single CPU or a set of CPUs, a set of CPUs being represented internally by `NDB` as a `SparseBitmask`. When attempting to lock to a set of CPUs, CPU usage was excessive due to the fact that the routine performing the locks used the `mt_thr_config.cpp::do_bind()` method, which looks for bits that are set over the entire theoretical range of the `SparseBitmask` (2^{32}-2, or 4294967294). This is fixed by using `SparseBitmask::getBitNo()`, which can be used to iterate over only those bits that are actually set, instead. (Bug #25799506)
- Setting `NoOfFragmentLogParts` such that there were more than 4 redo log parts per local data manager led to resource exhaustion and subsequent multiple data node failures. Since this is an invalid configuration, a check has been added to detect a configuration with more than 4 redo log parts per LDM, and reject it as invalid. (Bug #25333414)
- In certain cases, a failed `ALTER TABLE ... ADD UNIQUE KEY` statement could lead to SQL node failure. (Bug #24444878)

  References: This issue is a regression of: Bug #23089566.
- Error 240 is raised when there is a mismatch between foreign key trigger columns and the values supplied to them during trigger execution, but had no error message indicating the source of the problem. (Bug #23141739)

  References: See also: Bug #23068914, Bug #85857.
- If the number of LDM blocks was not evenly divisible by the number of TC/SPJ blocks, SPJ requests were not equally distributed over the available SPJ instances. Now a round-robin distribution is used to distribute SPJ requests across all available SPJ instances more effectively.

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

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

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

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

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

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

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

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

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

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

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

  References: See also: Bug #16275684.

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

- When an `NDB` table with foreign key constraints is created, its indexes are created first, and then, during foreign key creation, these indexes are loaded into the `NDB` dictionary cache. When a `CREATE TABLE` statement failed due to an issue relating to foreign keys, the indexes already in the cache were not

invalidated. This meant that any subsequent `CREATE TABLE` with any indexes having the same names as those in the failed statement produced inconsistent results. Now, in such cases, any indexes named in the failed `CREATE TABLE` are immediately invalidated from the cache. (Bug #85917, Bug #25882950)

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

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

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

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

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

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

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

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

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

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

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

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

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

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

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

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

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

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

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

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

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

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

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

    The `RESERVED` resource has been removed.

  - `IndexMemory` is no longer displayed in `ndb_config` output.

  (WL #9835, WL #10196)

- **Performance:** A number of debugging statements and printouts in the sources for the `DBTC` and `DBLQH` kernel blocks, as well as in related code, were moved into debugging code or removed altogether. This is expected to result in an improvement of up to 10% in the performance of local data management and transaction coordinator threads in many common use cases. (WL #10188)

- **NDB Cluster APIs; ndbinfo Information Database:** Added two tables to the `ndbinfo` information database. The `config_nodes` table provides information about nodes that are configured as part of a given NDB Cluster, such as node ID and process type. The `processes` table shows information

about nodes currently connected to the cluster; this information includes the process name and system process ID, and service address. For each data node and SQL node, it also shows the process ID of the node's angel process.

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

For more information, see The ndbinfo config_nodes Table, and The ndbinfo processes Table, as well as Ndb_cluster_connection::set_service_uri(). (WL #9819, WL #10147)

- **NDB Cluster APIs:** The system name of an NDB cluster is now visible in the `mysql` client as the value of the `Ndb_system_name` status variable, and can also be obtained by NDB API application using the `Ndb_cluster_connection::get_system_name()` method. The system name can be set using the `Name` parameter in the `[system]` section of the cluster configuration file. (WL #10321)

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

- Previously, when one LDM thread experienced I/O lag, such as during a disk overload condition, it wrote to a local checkpoint more slowly—that is, it wrote in I/O lag mode. However, other LDM threads did not necessarily observe or conform to this state. To ensure that write speed for the LCP is reduced by all LDM threads when such a slowdown is encountered, `NDB` now tracks I/O lag mode globally, so that I/O lag state is reported as soon as at least one thread is writing in I/O lag mode, and thus all LDM threads are forced to write in lag mode while the lag condition persists. This reduction in write speed by other LDM instances should increase overall capacity, enabling the disk overload condition to be overcome more quickly in such cases than before. (WL #10174)

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

  For more information, see ndb_import — Import CSV Data Into NDB. (WL #7614, WL #8862, WL #10653)

Bugs Fixed

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

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

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

- **Microsoft Windows:** When collecting information about CPUs on Windows, the Auto-Installer counted only physical cores, unlike on other platforms, where it collects information about both physical and

virtual cores. Now the CPU information obtained on Windows is the same as that provided on other platforms. (Bug #85209, Bug #25636423)

- **Solaris; ndbmemcache:** `ndbmemcache` was not built correctly on Solaris platforms when compiling NDB Cluster using Developer Studio. (Bug #85477, Bug #25730703)
- **Solaris; MySQL NDB ClusterJ:** ClusterJ was not built correctly on Solaris platforms when compiling NDB Cluster using Oracle Developer Studio. (Bug #25738510)
- **NDB Disk Data:** In some cases, setting dynamic in-memory columns of an NDB Disk Data table to `NULL` was not handled correctly. (Bug #79253, Bug #22195588)
- When `ndb_report_thresh_binlog_epoch_slip` was enabled, an event buffer status message with `report_reason=LOW/ENOUGH_FREE_EVENTBUFFER` was printed in the logs when event buffer usage was high and then decreased to a lower level. This calculation was based on total allocated event buffer memory rather than the limit set by `ndb_eventbuffer_max_alloc`; it was also printed even when the event buffer had unlimited memory (`ndb_eventbuffer_max_alloc` = 0, the default), which could confuse users.

  This is fixed as follows:

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

  (Bug #25726723)
- A bulk update is executed by reading records and executing a transaction on the set of records, which is started while reading them. When transaction initialization failed, the transaction executor function was subsequently unaware that this had occurred, leading to SQL node failures. This issue is fixed by providing appropriate error handling when attempting to initialize the transaction. (Bug #25476474)

  References: See also: Bug #20092754.
- CPU usage of the data node's main thread by the `DBDIH` master block as the end of a local checkpoint could approach 100% in certain cases where the database had a very large number of fragment replicas. This is fixed by reducing the frequency and range of fragment queue checking during an LCP. (Bug #25443080)
- Execution of an online `ALTER TABLE ... REORGANIZE PARTITION` statement on an `NDB` table having a primary key whose length was greater than 80 bytes led to restarting of data nodes, causing the reorganization to fail. (Bug #25152165)
- Multiple data node failures during a partial restart of the cluster could cause API nodes to fail. This was due to expansion of an internal object ID map by one thread, thus changing its location in memory, while another thread was still accessing the old location, leading to a segmentation fault in the latter thread.

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

  References: See also: Bug #25306089.

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

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

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

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

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

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

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

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

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

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

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

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

- There existed the possibility of a race condition between schema operations on the same database object originating from different SQL nodes; this could occur when one of the SQL nodes was late in

releasing its metadata lock on the affected schema object or objects in such a fashion as to appear to the schema distribution coordinator that the lock release was acknowledged for the wrong schema change. This could result in incorrect application of the schema changes on some or all of the SQL nodes or a timeout with repeated `waiting max ### sec for distributing...` messages in the node logs due to failure of the distribution protocol. (Bug #85010, Bug #25557263)

References: See also: Bug #24926009.

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

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

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

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

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

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

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

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

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

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

For more information, see Upgrading and Downgrading NDB Cluster. (WL #9778)

Bugs Fixed

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

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

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

  References: See also: Bug #18753341, Bug #23202735.
- **NDB Cluster APIs:** The adaptive send algorithm was not used as expected, resulting in every execution request being sent to the `NDB` kernel immediately, instead of trying first to collect multiple requests into larger blocks before sending them. This incurred a performance penalty on the order of 10%. The issue was due to the transporter layer always handling the *`forceSend`* argument used in several API methods (including `nextResult()` and `close()`) as `true`. (Bug #82738, Bug #24526123)
- The `ndb_print_backup_file` utility failed when attempting to read from a backup file when the backup included a table having more than 500 columns. (Bug #25302901)

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

  References: See also: Bug #25302901.
- No traces were written when `ndbmtd` received a signal in any thread other than the main thread, due to the fact that all signals were blocked for other threads. This issue is fixed by the removal of `SIGBUS`, `SIGFPE`, `SIGILL`, and `SIGSEGV` signals from the list of signals being blocked. (Bug #25103068)
- The `ndb_show_tables` utility did not display type information for hash maps or fully replicated triggers. (Bug #24383742)
- The NDB Cluster Auto-Installer did not show the user how to force an exit from the application (**CTRL**+**C**). (Bug #84235, Bug #25268310)

- The NDB Cluster Auto-Installer failed to exit when it was unable to start the associated service. (Bug #84234, Bug #25268278)
- The NDB Cluster Auto-Installer failed when the port specified by the `--port` option (or the default port 8081) was already in use. Now in such cases, when the required port is not available, the next 20 ports are tested in sequence, with the first one available being used; only if all of these are in use does the Auto-Installer fail. (Bug #84233, Bug #25268221)
- Multiples instances of the NDB Cluster Auto-Installer were not detected. This could lead to inadvertent multiple deployments on the same hosts, stray processes, and similar issues. This issue is fixed by having the Auto-Installer create a PID file (`mcc.pid`), which is removed upon a successful exit. (Bug #84232, Bug #25268121)
- When a data node running with `StopOnError` set to 0 underwent an unplanned shutdown, the automatic restart performed the same type of start as the previous one. In the case where the data node had previously been started with the `--initial` option, this meant that an initial start was performed, which in cases of multiple data node failures could lead to loss of data. This issue also occurred whenever a data node shutdown led to generation of a core dump. A check is now performed to catch all such cases, and to perform a normal restart instead.

  In addition, in cases where a failed data node was unable prior to shutting down to send start phase information to the angel process, the shutdown was always treated as a startup failure, also leading to an initial restart. This issue is fixed by adding a check to execute startup failure handling only if a valid start phase was received from the client. (Bug #83510, Bug #24945638)
- Data nodes that were shut down when the redo log was exhausted did not automatically trigger a local checkpoint when restarted, and required the use of `DUMP 7099` to start one manually. (Bug #82469, Bug #24412033)
- When a data node was restarted, the node was first stopped, and then, after a fixed wait, the management server assumed that the node had entered the `NOT_STARTED` state, at which point, the node was sent a start signal. If the node was not ready because it had not yet completed stopping (and was therefore not actually in `NOT_STARTED`), the signal was silently ignored.

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

  - Wait for data nodes to start shutting down (maximum 12 seconds)
  - Wait for data nodes to complete shutting down and reach `NOT_STARTED` state (maximum 120 seconds)

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

  References: See also: Bug #28728485.

Index

Symbols

--allow-pk-changes, 39, 104
--defaults-file, 58, 122
--diff-default, 71, 134
--disable-indexes, 30, 97

--ignore-extended-pk-changes, 39, 104
--initial, 51, 81, 116, 143
--ndb-log-fail-terminate, 39, 104
--nostart, 58, 122
--port, 81, 143
--query-all, 76, 138
--rebuild-indexes, 42, 107
--remap-column, 39, 104
--restore-epoch, 46, 112
--socket, 58, 122
-a, 76, 138
-flifetime-dse, 71, 134
.ctl files, 44, 81, 110, 143

A

aborted transactions, 58, 122
AbortOption, 39, 104
ACC scan, 51, 116
adaptive send, 81, 143
ADD DATAFILE, 48, 113
AES_ENCRYPT(), 10
ALTER, 42, 107
ALTER TABLE, 36, 63, 76, 102, 127, 138
ALTER TABLE ... ADD UNIQUE KEY, 71, 134
ANALYZE TABLE, 51, 116
ANTLR, 39, 104
AnyValue, 63
API nodes, 18, 76, 87, 138
API timeouts, 8
ApiConnectRecord, 8
ApiFailureHandlingTimeout, 7
AppArmor, 8
apt, 71, 134
arrays, 26, 94
assert, 12
AttrInfo, 76, 138
AUTHENTICATION_PAM_LOG, 14
Auto-Installer, 63, 76, 81, 127, 138, 143
autoincrement, 48, 113
AUTO_INCREMENT, 63, 127
averages, 12

B

backup, 32, 36, 42, 46, 58, 71, 99, 102, 107, 122, 134
BACKUP, 44, 110
BackupLogBufferSize, 42, 107
backups, 81, 143
BatchByteSize, 71, 134
batched_key_access, 28, 95
binary log, 58, 71, 122
binary logging, 14, 84
BitmaskImpl::setRange(), 58, 122
BLOB, 31, 36, 46, 48, 51, 54, 98, 102, 113, 116, 118

blobs, 35, 101
BufferedOutputStream, 71, 134
BuildIndexThreads, 63, 127
bulk updates, 76, 138

C

case sensitivity, 36, 102
changes
  NDB Cluster, 83
character sets, 44, 110
checkpoints, 18, 57, 87, 121
close(), 48, 113
cloud (NDB), 63, 127
ClusterJPA, 71, 134
ClusterTransactionImpl, 58, 122
CMake 4, 6
CMake3, 46, 112
CM_REGREF, 30, 97
COLUMN_FORMAT, 46, 112
comments, 6
compiling, 6, 46, 57, 63, 71, 76, 112, 121, 127, 134, 138
computeXorChecksum(), 26, 94
concurrency, 71, 134
concurrent operations, 48, 113
concurrent trigger operations, 54, 118
condition pushdown, 33, 51, 100, 116
config.ini, 58, 122
configuration, 48, 63, 71, 113, 127, 134
config_nodes table, 76, 138
conflict resolution, 16, 86
connection string, 54, 118
connection timeouts, 16, 86
connections, 18, 87
CONTINUEB signal, 28, 95
CONTRIBUTING.md, 8
CopyFrag, 22, 91
CopyGCI, 8
correlation IDs, 48, 113
CPU, 76, 138
CPU binding of the receive threads, 63, 127
CPU locking, 63, 127
CPU usage, 71, 134
cpustat, 48, 113
CREATE NODEGROUP, 31, 98
CREATE TABLE, 57, 76, 121
CSV, 39, 76, 138

D

data files, 81, 143
data memory, 71, 134
data node, 54, 63, 76, 118, 127, 138
data node failure, 71, 134
data node failures, 76, 138

data node shutdown, 62, 126
data nodes, 20, 22, 25, 58, 62, 71, 76, 89, 91, 93, 122, 126, 134, 138
data pages, 44, 110
DataMemory, 44, 71, 76, 110, 134, 138
Date, 39, 104
DBACC, 63, 71, 76, 127, 134, 138
Dbacc::getLastAndRemove(), 63, 127
Dbacc::startNext(), 54, 118
DBDICT, 20, 28, 89, 95
DBDIH, 33, 76, 100, 138
DBLQH, 63, 76, 127, 138
DBSPJ, 8, 20, 46, 48, 63, 71, 76, 89, 112, 113, 127, 134, 138
DBTC, 8, 14, 18, 46, 58, 63, 71, 76, 84, 87, 112, 122, 127, 134, 138
Dbtc::execSIGNAL_DROPPED_REP(), 76, 138
DBTUP, 24, 54, 63, 71, 76, 93, 118, 127, 134, 138
DBTUX, 63, 127
DELETE, 31, 71, 98, 134
DELETE_WAIT, 51, 116
Developer Studio, 76, 138
dictionary cache, 71, 134
DIGETNODESREF, 76, 138
disconnection, 51, 116
discrete values, 63, 127
disk format, 42, 107
disk overload, 76, 138
DiskFree, 63, 127
DISK_RECORDS, 20, 89
DIVERIFYREQ, 28, 95
dojo, 35, 36, 101, 102
dojo.zip, 57, 121
DROP DATABASE, 51
DROP INDEX, 10
DROP NODEGROUP, 22, 91
DROP TABLE, 51, 57, 71, 116, 121, 134
DROP_TRIG_IMPL_REQ, 54, 118
DUMP, 44, 110
DUMP 11001, 28, 95
DUMP 7027, 63, 127
DUMP 7099, 81, 143
DUMP 9988, 42, 107
DUMP 9989, 42, 107
DUMP codes, 63, 127
DumpPageMemoryOnFail, 36, 102
duplicate weedout, 63, 127
DYNAMIC, 76, 138
dynamic index memory, 63, 127
DynArr256::set(), 71, 134

E

element deletion, 63, 127
EnablePartialLcp, 63, 127
EnableRedoControl, 36, 57, 102, 121
epochs, 46, 112

error 240, 71, 134
error 631, 71, 134
error codes, 54, 118
error handling, 31, 63, 71, 98, 127, 134
error log, 31, 98
error messages, 44, 63, 110, 127
errors, 16, 24, 46, 71, 76, 86, 93, 112, 134, 138
ER_NO_REFERENCED_ROW_2, 54, 118
event buffer, 32, 99
EventBuffer, 12, 84
EventBytesRecvdCount, 26, 94
execute_signals(), 71, 134
exit, 81, 143
EXPLAIN, 63, 76, 127, 138

F

files, 28, 95
FIXED, 46, 112
FLUSH_AI, 76, 138
foreign keys, 20, 42, 54, 58, 71, 76, 89, 107, 118, 122, 134, 138
free memory, 12, 84
free pages, 71, 134
FTS, 10
FULLY REPLICATED, 35, 101
FULLY_REPLICATED, 81, 143

G

gcc, 71, 134
GCI, 14, 36, 54, 84, 102, 118
GCI boundary, 58, 122
GCP, 16, 42, 86, 107
GCP stall, 22, 91
GCP stalls, 16, 86
GCP_SAVEREQ, 36, 102
getColumn(), 71, 134
getCreateSchemaVersion(), 44, 110
getGCIEventOperations(), 20, 89
GET_TABLEID_REQ, 30, 97
glibc, 63, 127
global checkpoints, 20, 33, 89, 100
global schema lock, 44, 110
GOUP BY, 71, 134
GSN_TRANSID_AI, 25, 93

H

HashMap, 81, 143
ha_ndbcluster::copy_fk_for_offline_alter(), 58, 122
ha_ndbcluster::exec_bulk_update(), 76, 138
heartbeat failure handling, 71, 134
host_info, 57, 121

I

I/O lag, 76, 138

identifiers, 7
idxbld, 63, 127
IF NOT EXISTS, 57, 121
Important Change, 6, 10, 18, 25, 26, 28, 42, 46, 51, 58, 63, 71, 76, 87, 93, 94, 95, 107, 112, 116, 122, 127, 134, 138
IN, 58, 122
IN(), 18, 87
Incompatible Change, 42, 63, 76, 107, 127, 138
index invalidation, 71, 134
index length, 51, 116
index statistics, 10, 20, 22, 24, 44, 89, 91, 93, 110
IndexMemory, 71, 76, 134, 138
INFORMATION_SCHEMA.TABLES, 51, 116
InitFragmentLogFiles, 63, 127
InitialNoOfOpenFiles, 48, 113
INNER JOIN, 58, 122
InnoDB, 6, 7, 8, 12, 16
insert operations, 54, 118
InsertRecoveryWork, 57, 121
InstallDirectory, 62, 126
invalid configuration, 71, 134

J

jam(), 76, 138
Java 11 deprecation warnings, 39, 104
Java versions, 39, 104
job buffer, 63, 71, 127, 134
job buffer full, 54, 118
joins, 18, 87
JOIN_TAB, 63, 127

L

LCP, 32, 42, 44, 51, 54, 57, 62, 63, 71, 76, 81, 99, 107, 110, 116, 118, 121, 126, 127, 134, 138, 143
LCP pause, 51, 116
LCP protocol, 63, 127
LCP_COMPLETE_REP, 42, 48, 63, 107, 113, 127
LCP_FRAG_REP, 76, 138
LCP_SCANNED_BIT, 22, 91
LCP_STATUS_IDLE, 48, 113
LCP_TAB_SAVED, 44, 110
LDM, 28, 42, 71, 76, 95, 107, 134, 138
LGMAN, 63, 127
libclass-methodmaker-perl, 71, 134
libndbclient-devel, 54, 118
limitations (removal), 71, 134
loading data, 76, 138
LocationDomainId, 63, 127
lock contention, 48, 113
locks, 42, 107
log buffer, 58, 122
log files, 71, 134
LogBuffer, 71, 134

logbuffers, 58, 122
logging, 18, 28, 39, 42, 63, 71, 87, 95, 104, 107, 127, 134
long signals, 46, 76, 112, 138
LongMessageBuffer, 76, 138
LONGVARBINARY, 58, 122
lookups, 71, 134
LooseScan, 63, 127
LQHKEYREQ, 46, 112
LQH_TRANSCONF, 7

M

MASTER_LCPCONF, 46, 112
MASTER_LCP_REQ, 46, 112
MASTER_POS_WAIT(), 7
materialized semijoin, 63, 127
MaxBufferedEpochs, 48, 113
MaxDiskWriteSpeedOwnRestart, 36, 102
MaxFKBuildBatchSize, 63, 127
MaxNoOfExecutionThreads, 51, 58, 71, 116, 122, 134
MaxNoOfOpenFiles, 48, 113
MaxNoOfOrderedIndexes, 51, 116
MaxNoOfTables, 51, 116
MaxNoOfUniqueHashIndexes, 51, 116
maxRecordSize, 71, 134
MaxReorgBuildBatchSize, 63, 127
MaxUIBuildBatchSize, 63, 127
MAX_EXECUTION_TIME, 48, 113
MAX_ROWS, 44, 110
mcc.pid, 76, 138
MD5() encryption function, 14
memory exhaustion, 71, 134
memory usage, 36, 102
metadata, 58, 122
metadata lock, 76, 138
mgm client commands, 63, 127
MgmtSrvr::restartNodes(), 81, 143
Microsoft Windows, 14, 20, 42, 76, 84, 89, 107, 138
mt.cpp, 71, 134
mt_thr_config.cpp::do_bind(), 71, 134
MySQL Enterprise Monitor, 28, 95
MySQL NDB ClusterJ, 10, 20, 22, 39, 44, 51, 54, 57, 58, 63, 71, 76, 89,
91, 104, 110, 116, 118, 121, 122, 127, 134, 138
mysqld, 32, 48, 51, 71, 76, 99, 113, 116, 134, 138
mysqldump, 7, 10, 12
my_print_help(), 8
m_buffer, 71, 134
m_buffered_size, 71, 134
m_max_batch_size_bytes, 76, 138
m_sending, 71, 134
m_sending_size, 71, 134

N

NDB Client Programs, 12, 12, 16, 33, 35, 39, 54, 57, 58, 62, 63, 84, 86,
100, 101, 104, 118, 121, 122, 126, 127
NDB Cluster, 5, 6, 7, 8, 10, 12, 12, 14, 16, 18, 20, 22, 24, 25, 26,
28, 30, 31, 32, 33, 35, 36, 39, 42, 44, 46, 48, 51, 54, 57, 58, 62,
63, 71, 76, 81, 84, 84, 86, 87, 89, 91, 93, 93, 94, 95, 97, 98, 99,
100, 101, 102, 104, 107, 110, 112, 113, 116, 118, 121, 122, 126, 127, 134, 138, 143
NDB Cluster APIs, 8, 10, 16, 18, 20, 25, 28, 31, 35, 48, 51, 58, 71, 76,
81, 86, 87, 89, 93, 95, 98, 101, 113, 116, 122, 134, 138, 143
NDB Disk Data, 36, 42, 44, 48, 51, 54, 63, 71, 76, 81, 102, 107, 110, 113,
116, 118, 127, 134, 138, 143
NDB Operator, 14, 84
NDB Replication, 16, 25, 33, 46, 51, 63, 71, 76, 93
NDB$BLOB, 51
ndb-batch-size, 26, 94
ndb-common, 31, 98
ndb-update-minimal, 71
Ndb::getNextEventOpInEpoch3(), 63
Ndb::pollEvents2(), 16, 86
ndbcluster_print_error(), 48, 113
ndbd, 12, 26, 28, 81, 94, 95, 143
ndberr, 54, 118
NdbEventBuffer, 26, 94
NDBFS, 16, 63, 127
NdbfsDumpRequests, 48, 113
ndbimport, 39
NdbIndexScanOperation::setBound(), 58, 122
ndbinfo, 28, 63, 95, 127
ndbinfo Information Database, 8, 57, 63, 76, 121, 127, 138
ndbinfo.cluster_locks, 31, 98
ndbinfo.restart_info, 14, 84
NdbInterpretedCode, 18, 87
NDBJTie, 71, 134
ndbmemcache, 33, 35, 76, 100, 101, 138
ndbmtd, 36, 48, 54, 71, 76, 81, 102, 113, 118, 134, 138, 143
NdbObjectIdMap, 76, 138
NdbOperation, 18, 87
ndbout, 54, 118
NdbReceiver, 58, 122
NdbReceiverBuffer, 39, 104
NdbRecord, 10
NdbScanFilter, 51, 116
NdbScanOperation, 58, 122
NDBT, 46, 112
NdbTable, 71, 134
NdbThread_SetThreadPrio, 25, 93
NdbTransaction, 48, 113
ndb_binlog_index, 51, 116
ndb_blob_tool, 39, 104
Ndb_cluster_connection::get_system_name(), 76, 138
Ndb_cluster_connection::set_service_uri(), 76, 138
ndb_config, 51, 71, 76, 116, 134, 138
ndb_desc, 32, 99

ndb_eventbuffer_max_alloc, 76, 138
ndb_import, 26, 28, 31, 36, 54, 76, 94, 95, 98, 102, 118, 138
NDB_LE_EventBufferStatus, 31, 98
Ndb_logevent_type, 31, 98
ndb_log_update_as_write, 16
ndb_log_update_minimal, 16
ndb_mgmd, 20, 22, 28, 35, 54, 62, 89, 91, 95, 101, 118, 126
NDB_MGM_NODE_TYPE_UNKNOWN, 30, 97
ndb_perror, 63, 127
ndb_print_backup_file, 81, 143
ndb_redo_log_reader, 12, 84
ndb_report_thresh_binlog_epoch_slip, 76, 138
Ndb_rep_tab_key, 31, 98
ndb_restore, 12, 16, 20, 30, 31, 33, 35, 36, 39, 42, 44, 46, 48, 51, 58,
81, 89, 97, 98, 100, 101, 102, 104, 107, 110, 112, 113, 116, 122, 143
ndb_row_checksum, 54, 118
ndb_select_all, 16, 86
ndb_setup.py, 33, 35, 57, 100, 101, 121
ndb_show_tables, 39, 71, 81, 104, 134, 143
Ndb_system_name, 76, 138
NDB_TABLE, 36, 63, 102, 127
ndb_top, 58, 63, 71, 122, 127, 134
Ndb_UnlockCPU(), 63, 127
ndb_waiter, 20, 39, 89, 104
node failure, 71, 76, 134, 138
node failure handling, 26, 33, 63, 71, 94, 100, 127, 134
node failures, 71, 134
node ID allocation, 44, 110
node IDs, 26, 94
node recovery, 71, 134
node restart, 63, 127
node restarts, 36, 81, 102, 143
node shutdown, 10, 22, 91
node takeover, 44, 110
Node.js, 35, 101
NodeGroup, 35, 101
NODELOG DEBUG, 63, 127
NoOfFragmentLogParts, 71, 134
NOT_STARTED, 81, 143
NOWAIT, 18, 87
NO_OF_BUCKETS, 30
NULL, 26, 51, 58, 76, 94, 116, 122, 138
num-slices, 42, 107

O

ODirect, 63, 127
ODirectSyncFlag, 63, 127
offline_mode, 12
OM_CREATE, 63, 127
OM_WRITE_BUFFER, 63, 127
ON DELETE CASCADE, 48, 113
online operations, 12, 76, 84, 138
online table reorganization, 14, 84

OpenSSL, 18, 87
open_table(), 57, 121
OPTIMIZE TABLE, 12
options, 58, 122
ORDER BY, 46, 112
O_DIRECT, 35, 42, 63, 101, 107, 127
O_SYNC, 63, 127

P

packaging, 8, 10, 71
Packaging, 31, 35, 51, 54, 71, 81, 98, 101, 116, 118, 134, 143
parallelism, 63, 127
Paramiko, 63, 127
partial LCP, 42, 63, 107, 127
partial restarts, 42, 107
PartionCount, 32, 99
Partitioning, 76, 138
PARTITION_BALANCE, 58, 122
password, 63, 127
Performance, 26, 54, 76, 94, 118, 138
perror --ndb, 63, 127
PGMAN, 42, 63, 107, 127
PID, 81, 143
PK read, 51, 116
poll_owner, 81, 143
PRIMARY KEY, 76, 138
primary key updates, 16
primary keys, 16, 18, 87
processes, 20, 57, 89, 121
processes table, 76, 138
pushdown joins, 71, 134
pushed join, 63, 127
pushed joins, 46, 112
pycrypto, 81, 143
python-paramiko, 71, 134

Q

QEP_TAB, 63, 127
QMGR, 7, 63, 127
query cache, 76, 138
query memory, 48, 113

R

race, 58, 122
race condition, 48, 113
range checks, 28, 95
range scans, 28, 95
readln_socket(), 14, 84
READ_BACKUP, 24, 93
read_cost(), 28, 95
read_length, 76, 138
realtime break, 20, 89
RealtimeScheduler, 35, 101

receive thread, 26, 58, 94, 122
receive thread activation threshold, 63, 127
reconnection, 71, 134
RecoveryWork, 57, 63, 121, 127
redo log, 22, 32, 33, 48, 57, 63, 81, 91, 99, 100, 113, 121, 127, 143
redo log file rotation, 71, 134
redo log part metadata, 63, 127
RedoOverCommitCounter, 42, 107
RedoOverCommitLimit, 42, 107
REGCONF, 18, 87
REGREQ, 18, 87
REORGANIZE PARTITION, 22, 62, 76, 91, 126, 138
Replication, 7, 63
replica_allow_batching, 25, 93
request distribution, 71, 134
RESET REPLICA, 33
RESET SLAVE, 33
resource allocation, 44, 110
RESTART, 51, 116
restarts, 44, 57, 58, 63, 71, 76, 110, 121, 122, 127, 134, 138
restore, 54, 57, 118, 121
result buffers, 71, 134
rolling restart, 14, 84
ROLLUP, 8
row ID, 51, 116

S

SafeCounter, 48, 113
scaling, 76, 138
scanIndex(), 58, 122
SCANREQ, 71, 134
scans, 8, 54, 81, 118, 143
SCANTABREQ, 36, 102
SCAN_FRAGCONF, 63, 127
SCAN_FRAGREF, 63, 127
SCAN_FRAGREQ, 54, 63, 118, 127
SCAN_NEXTREQ, 6
schema distribution coordinator, 76, 138
schema operations, 76, 138
SCI, 58, 122
scratch buffer, 20, 89
SECURITY.md, 8
SELECT, 42, 51, 107, 116
semijoin, 63, 127
send buffer, 58, 81, 122, 143
send buffers, 58, 81, 122, 143
send_buffer::m_node_total_send_buffer_size, 71, 134
ServerPort, 62, 126
service, 81, 143
SessionFactory, 71, 134
SET_LOGLEVELORD, 28, 95
SharedGlobalMemory, 46, 112
SHM, 10, 58, 122

SHOW CREATE TABLE, 76, 138
SHUTDOWN, 76, 138
shutdown, 76, 138
signal data, 30, 97
signal dump, 7
signal memory, 33, 100
signals, 22, 25, 58, 81, 91, 93, 122, 143
SignalSender, 42, 107
SIGTERM, 20, 32, 89, 99
singal IDs, 20, 89
slave SQL thread, 76
slave_parallel_workers, 71
slice-id, 42, 107
SNAPSHOTSTART, 46
Solaris, 6, 76, 138
SparseBitmask::getBitNo(), 71, 134
SPJ, 28, 58, 63, 71, 95, 122, 127, 134
SQL nodes, 31, 98
SSH, 63, 127
SSL, 10
START_LCP_REQ, 42, 107
statistics, 26, 32, 94, 99
stdscr, 63, 127
STOP, 48, 113
stop GCI, 58, 122
StopOnError, 81, 143
storage format, 63, 127
STRAIGHT_JOIN, 46, 112
subqueries, 8
SUB_GCP_COMPLETE_ACK, 81, 143
SUB_GCP_COMPLETE_REP, 51, 116
SUB_STOP_REQ, 54, 118
SUMA, 22, 24, 30, 54, 91, 93, 97, 118
sysfiles, 44, 110
SYSTAB_0, 48, 113
system restart, 71, 134
system restarts, 8
system tables, 28, 95
sysvar_audit_log_strategy, 10

T

table reorganization, 28, 44, 95, 110
Table::getColumn(), 35, 101
tableno, 63, 127
tablespace full, 71, 134
Table_Map, 63
TABLE_READ_ONLY, 44, 110
TAB_SCANREF, 24, 93
takeover, 46, 112
TC, 16, 86
TCGETOPSIZEREQ, 63, 127
TCKEYREQ, 46, 112
TCRELEASEREQ, 8

TC_COMMIT_ACK, 81, 143
TEXT, 48, 54, 113, 118
TE_ALTER, 51, 116
thread priority, 26, 94
ThreadConfig, 51, 63, 116, 127
timeout, 76, 138
timeouts, 81, 143
TINYBLOB, 46, 112
trace logs, 20, 89
traces, 81, 143
transaction coordinator, 12, 44, 110
transactions, 26, 42, 94, 107
TRANSID_AI, 71, 76, 134, 138
TransporterRegistry, 16, 86
TransporterRegistry::prepareSendTemplate(), 58, 122
transporters, 16, 86
TRANS_AI, 76, 138
triggers (NDB), 54, 118
TRIX, 10
TRUNCATE, 63, 127
TSMAN, 63, 127
TUP scan, 51, 116
tuple corruption, 54, 118
TwoPassInitialNodeRestartCopy, 54, 63, 118, 127
type conversion, 46, 112

U

Ubuntu, 14, 84
undo files, 51, 116
undo log, 31, 63, 98, 127
undo log files, 81, 143
unique keys, 71, 134
unit tests, 71, 134
unplanned shutdown, 71, 81, 134, 143
unqualified option, 71, 134
UPDATE CASCADE, 76, 138
upgrades, 36, 44, 46, 48, 102, 110, 112, 113
upgrades and downgrades, 81, 143
UseShm, 58, 122
UTF8, 7

V

VIRTUAL, 46
Visual Studio, 14, 84

W

wait locks, 54, 118
WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS(), 6
warnings, 12, 18, 84, 87
watchdog, 7
Windows, 63, 127
WITH_UNIT_TESTS, 63, 127
worker threads, 71, 134