MySQL NDB Cluster 7.5 Release Notes

Abstract

This document contains release notes for the changes in each release of MySQL NDB Cluster that uses version 7.5 of the NDB (NDBCLUSTER) storage engine.

Each NDB Cluster 7.5 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 7.5, see What is New in NDB Cluster 7.5. For a complete list of all bug fixes and feature changes in MySQL Cluster, please refer to the changelog section for each individual NDB Cluster release.

For additional MySQL 5.7 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: 2025-04-16 (revision: 29972)

Table of Contents

Preface and Legal Notices	პ
Changes in MySQL NDB Cluster 7.5.36 (5.7.44-ndb-7.5.36) (2024-10-15, General Availability)	5
Changes in MySQL NDB Cluster 7.5.35 (5.7.44-ndb-7.5.35) (2024-07-02, General Availability)	5
Changes in MySQL NDB Cluster 7.5.34 (5.7.44-ndb-7.5.34) (2024-04-30, General Availability)	6
Changes in MySQL NDB Cluster 7.5.33 (5.7.44-ndb-7.5.33) (2024-01-16, General Availability)	7
Changes in MySQL NDB Cluster 7.5.32 (5.7.44-ndb-7.5.32) (2023-10-26, General Availability)	7
Changes in MySQL NDB Cluster 7.5.31 (5.7.43-ndb-7.5.31) (2023-07-18, General Availability)	8
Changes in MySQL NDB Cluster 7.5.30 (5.7.42-ndb-7.5.30) (2023-04-19, General Availability)	9
Changes in MySQL NDB Cluster 7.5.29 (5.7.41-ndb-7.5.29) (2023-01-18, General Availability)	. 11
Changes in MySQL NDB Cluster 7.5.28 (5.7.40-ndb-7.5.28) (2202-10-12, General Availability)	. 13
Changes in MySQL NDB Cluster 7.5.27 (5.7.39-ndb-7.5.27) (2022-07-27, General Availability)	. 14
Changes in MySQL NDB Cluster 7.5.26 (5.7.38-ndb-7.5.26) (2022-04-27, General Availability)	. 14
Changes in MySQL NDB Cluster 7.5.25 (5.7.37-ndb-7.5.25) (2022-01-19, General Availability)	. 16
Changes in MySQL NDB Cluster 7.5.24 (5.7.36-ndb-7.5.24) (2021-10-20, General Availability)	
Changes in MySQL NDB Cluster 7.5.23 (5.7.35-ndb-7.5.23) (2021-07-21, General Availability)	. 19
Changes in MySQL NDB Cluster 7.5.22 (5.7.34-ndb-7.5.22) (2021-04-21, General Availability)	. 20
Changes in MySQL NDB Cluster 7.5.21 (5.7.33-ndb-7.5.21) (2021-01-19, General Availability)	. 21
Changes in MySQL NDB Cluster 7.5.20 (5.7.32-ndb-7.5.20) (2020-10-20, General Availability)	. 22
Changes in MySQL NDB Cluster 7.5.19 (5.7.31-ndb-7.5.19) (2020-07-14, General Availability)	. 23
Changes in MySQL NDB Cluster 7.5.18 (5.7.30-ndb-7.5.18) (2020-04-28, General Availability)	. 25

Changes in MySQL NDB Cluster 7.5.17 (5.7.29-ndb-7.5.17) (2020-01-14, General Availability)	
Changes in MySQL NDB Cluster 7.5.16 (5.7.28-ndb-7.5.16) (2019-10-15, General Availability)	
Changes in MySQL NDB Cluster 7.5.15 (5.7.27-ndb-7.5.15) (2019-07-23, General Availability)	
Changes in MySQL NDB Cluster 7.5.14 (5.7.26-ndb-7.5.14) (2019-04-26, General Availability)	
Changes in MySQL NDB Cluster 7.5.13 (5.7.25-ndb-7.5.13) (2019-01-22, General Availability)	
Changes in MySQL NDB Cluster 7.5.12 (5.7.24-ndb-7.5.12) (2018-10-23, General Availability)	35
Changes in MySQL NDB Cluster 7.5.11 (5.7.23-ndb-7.5.11) (2018-07-27, General Availability)	36
Changes in MySQL NDB Cluster 7.5.10 (5.7.22-ndb-7.5.10) (2018-04-20, General Availability)	38
Changes in MySQL NDB Cluster 7.5.9 (5.7.21-ndb-7.5.9) (2018-01-17, General Availability)	
Changes in MySQL NDB Cluster 7.5.8 (5.7.20-ndb-7.5.8) (2017-10-18, General Availability)	
Changes in MySQL NDB Cluster 7.5.7 (5.7.19-ndb-7.5.7) (2017-07-19, General Availability)	
Changes in MySQL NDB Cluster 7.5.6 (5.7.18-ndb-7.5.6) (2017-04-10, General Availability)	
Changes in MySQL NDB Cluster 7.5.5 (5.7.17-ndb-7.5.5) (2017-01-17, General Availability)	
Changes in MySQL NDB Cluster 7.5.3 (5.7.17-1db-7.5.4) (2016-10-18, General Availability)	
Changes in MySQL NDB Cluster 7.5.3 (5.7.10-hdb-7.5.3) (2016-07-08, Release Candidate)	
Changes in MySQL NDB Cluster 7.5.2 (5.7.12-ndb-7.5.2) (2016-06-01, Development Milestone)	
Changes in MySQL NDB Cluster 7.5.1 (5.7.11-ndb-7.5.1) (2016-03-18, Development Milestone)	
Changes in MySQL NDB Cluster 7.5.0 (5.7.10-ndb-7.5.0) (2016-02-05, Development Milestone)	
Release Series Changelogs: MySQL NDB Cluster 7.5	79
Changes in MySQL NDB Cluster 7.5.34 (5.7.44-ndb-7.5.34) (2024-04-30, General	
Availability)	80
Changes in MySQL NDB Cluster 7.5.33 (5.7.44-ndb-7.5.33) (2024-01-16, General	
Availability)	80
Changes in MySQL NDB Cluster 7.5.31 (5.7.43-ndb-7.5.31) (2023-07-18, General	
Availability)	80
Changes in MySQL NDB Cluster 7.5.30 (5.7.42-ndb-7.5.30) (2023-04-19, General	
Availability)	81
Changes in MySQL NDB Cluster 7.5.29 (5.7.41-ndb-7.5.29) (2023-01-18, General	01
Availability)	83
Changes in MySQL NDB Cluster 7.5.28 (5.7.40-ndb-7.5.28) (2202-10-12, General	03
	0.4
Availability)	84
Changes in MySQL NDB Cluster 7.5.27 (5.7.39-ndb-7.5.27) (2022-07-27, General	-
Availability)	85
Changes in MySQL NDB Cluster 7.5.26 (5.7.38-ndb-7.5.26) (2022-04-27, General	
Availability)	85
Changes in MySQL NDB Cluster 7.5.25 (5.7.37-ndb-7.5.25) (2022-01-19, General	
Availability)	86
Changes in MySQL NDB Cluster 7.5.24 (5.7.36-ndb-7.5.24) (2021-10-20, General	
Availability)	88
Changes in MySQL NDB Cluster 7.5.23 (5.7.35-ndb-7.5.23) (2021-07-21, General	
Availability)	89
Changes in MySQL NDB Cluster 7.5.22 (5.7.34-ndb-7.5.22) (2021-04-21, General	-
Availability)	90
Changes in MySQL NDB Cluster 7.5.21 (5.7.33-ndb-7.5.21) (2021-01-19, General	00
Availability)	00
Changes in MySQL NDB Cluster 7.5.20 (5.7.32-ndb-7.5.20) (2020-10-20, General	90
	~4
Availability)	91
Changes in MySQL NDB Cluster 7.5.19 (5.7.31-ndb-7.5.19) (2020-07-14, General	
Availability)	92
Changes in MySQL NDB Cluster 7.5.18 (5.7.30-ndb-7.5.18) (2020-04-28, General	
Availability)	94
Changes in MySQL NDB Cluster 7.5.17 (5.7.29-ndb-7.5.17) (2020-01-14, General	
Availability)	95
Changes in MySQL NDB Cluster 7.5.16 (5.7.28-ndb-7.5.16) (2019-10-15, General	
Availability)	96
Changes in MySQL NDB Cluster 7.5.15 (5.7.27-ndb-7.5.15) (2019-07-23, General	_
Availability)	98
· · ·	-50

	Changes in MySQL NDB Cluster 7.5.14 (5.7.26-ndb-7.5.14) (2019-04-26, General Availability)	. 99
	Changes in MySQL NDB Cluster 7.5.13 (5.7.25-ndb-7.5.13) (2019-01-22, General	
	Availability)	101
	Changes in MySQL NDB Cluster 7.5.12 (5.7.24-ndb-7.5.12) (2018-10-23, General	400
	Availability)	102
	Availability)	104
	Changes in MySQL NDB Cluster 7.5.10 (5.7.22-ndb-7.5.10) (2018-04-20, General	104
	Availability)	105
	Changes in MySQL NDB Cluster 7.5.9 (5.7.21-ndb-7.5.9) (2018-01-17, General Availability) .	
	Changes in MySQL NDB Cluster 7.5.8 (5.7.20-ndb-7.5.8) (2017-10-18, General Availability).	107
	Changes in MySQL NDB Cluster 7.5.7 (5.7.19-ndb-7.5.7) (2017-07-19, General Availability) .	
	Changes in MySQL NDB Cluster 7.5.6 (5.7.18-ndb-7.5.6) (2017-04-10, General Availability) .	
	Changes in MySQL NDB Cluster 7.5.5 (5.7.17-ndb-7.5.5) (2017-01-17, General Availability) .	
	Changes in MySQL NDB Cluster 7.5.4 (5.7.16-ndb-7.5.4) (2016-10-18, General Availability) .	
	Changes in MySQL NDB Cluster 7.5.3 (5.7.13-ndb-7.5.3) (2016-07-08, Release Candidate)	125
	Changes in MySQL NDB Cluster 7.5.2 (5.7.12-ndb-7.5.2) (2016-06-01, Development	407
	Milestone)	127
	Milestone)	133
	Changes in MySQL NDB Cluster 7.5.0 (5.7.10-ndb-7.5.0) (2016-02-05, Development	133
	Milestone)	136
es	,	143

Preface and Legal Notices

This document contains release notes for the changes in each release of MySQL NDB Cluster that uses version 7.5 of the NDB storage engine.

Legal Notices

Copyright © 1997, 2025, 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 errorfree. 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 7.5.36 (5.7.44-ndb-7.5.36) (2024-10-15, General Availability)

MySQL NDB Cluster 7.5.36 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).



Note

NDB 7.5.36 is the final release in the NDB Cluster 7.5 series.

Users of NDB 7.5 are encouraged to upgrade to the latest NDB 8.0, 8.4, or Innovation release.

Bugs Fixed

- 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 server did not always handle connections correctly when running with both the thread pool and audit log plugins. (Bug #36682079)
- 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 7.5.35 (5.7.44-ndb-7.5.35) (2024-07-02, General Availability)

MySQL NDB Cluster 7.5.35 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

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.

Changes in MySQL NDB Cluster 7.5.34 (5.7.44-ndb-7.5.34) (2024-04-30, General Availability)

MySQL NDB Cluster 7.5.34 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the $_{\rm NDB}$ storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

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.

• Removed an assertion failure in sql/field.cc. (Bug #112503, Bug #35846221)

Changes in MySQL NDB Cluster 7.5.33 (5.7.44-ndb-7.5.33) (2024-01-16, General Availability)

MySQL NDB Cluster 7.5.33 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

- Compilation Notes
- · Pluggable Authentication
- Bugs Fixed

Compilation Notes

• **Microsoft Windows:** NDB Cluster did not compile correctly using Visual Studio 2022. (Bug #35967676)

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

• In limited cases, passing data to the MD5 () encryption function could halt the server. (Bug #35764496)

Changes in MySQL NDB Cluster 7.5.32 (5.7.44-ndb-7.5.32) (2023-10-26, General Availability)

MySQL NDB Cluster 7.5.32 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

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)

Changes in MySQL NDB Cluster 7.5.31 (5.7.43-ndb-7.5.31) (2023-07-18, General Availability)

MySQL NDB Cluster 7.5.31 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

- · Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

• NDB 7.5 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 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)

Changes in MySQL NDB Cluster 7.5.30 (5.7.42-ndb-7.5.30) (2023-04-19, General Availability)

MySQL NDB Cluster 7.5.30 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

- · 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)
- 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)

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 #90360, Bug
#27834961)

References: See also: Bug #85411, Bug #25703113.

Changes in MySQL NDB Cluster 7.5.29 (5.7.41-ndb-7.5.29) (2023-01-18, General Availability)

MySQL NDB Cluster 7.5.29 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

- Compilation Notes
- · Bugs Fixed

Compilation Notes

Added GCC 12 support. (Bug #33574176)

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)
- 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 7.5.28 (5.7.40-ndb-7.5.28) (2202-10-12, General Availability)

MySQL NDB Cluster 7.5.28 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

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 7.5.27 (5.7.39-ndb-7.5.27) (2022-07-27, General Availability)

MySQL NDB Cluster 7.5.27 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

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 7.5.26 (5.7.38-ndb-7.5.26) (2022-04-27, General Availability)

MySQL NDB Cluster 7.5.26 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the \mathtt{NDB} storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

- · 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 <code>Dbacc::getElement()</code>, and the root cause identified to be the timing of resizing the structures used for storing elements by <code>DBACC</code>, 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 <code>computeXorChecksum()</code> 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 <code>xor</code> 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 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.

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 7.5.25 (5.7.37-ndb-7.5.25) (2022-01-19, General Availability)

MySQL NDB Cluster 7.5.25 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

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)
- 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)
- 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.

- 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 7.5.24 (5.7.36-ndb-7.5.24) (2021-10-20, General Availability)

MySQL NDB Cluster 7.5.24 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

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 7.5.23 (5.7.35-ndb-7.5.23) (2021-07-21, General Availability)

MySQL NDB Cluster 7.5.23 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

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

Changes in MySQL NDB Cluster 7.5.22 (5.7.34-ndb-7.5.22) (2021-04-21, General Availability)

MySQL NDB Cluster 7.5.22 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

Bugs Fixed

- 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)
- 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)

Changes in MySQL NDB Cluster 7.5.21 (5.7.33-ndb-7.5.21) (2021-01-19, General Availability)

MySQL NDB Cluster 7.5.21 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

- Deprecation and Removal Notes
- Bugs Fixed

Deprecation and Removal Notes

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

References: See also: Bug #31888835.

• **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)
- 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 7.5.20 (5.7.32-ndb-7.5.20) (2020-10-20, General Availability)

MySQL NDB Cluster 7.5.20 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

- Deprecation and Removal Notes
- Bugs Fixed

Deprecation and Removal Notes

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

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

- NDB Client Programs: Effective with this release, the MySQL NDB Cluster Auto-Installer (ndb_setup.py) has been deprecated and is subject to removal in a future version of NDB Cluster. (Bug #31888835)
- **ndbmemcache:** ndbmemcache is deprecated in this release of NDB Cluster, and is scheduled for removal in the next release. (Bug #31876970)

Bugs Fixed

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

Changes in MySQL NDB Cluster 7.5.19 (5.7.31-ndb-7.5.19) (2020-07-14, General Availability)

MySQL NDB Cluster 7.5.19 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

Bugs Fixed

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

requests. While the copy operation is in progress, user-level SUB_START and SUB_STOP requests are blocked using a DICT lock.

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

- 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)

• 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)

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 7.5.18 (5.7.30-ndb-7.5.18) (2020-04-28, General Availability)

MySQL NDB Cluster 7.5.18 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

- · 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)

• 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 7.5.17 (5.7.29-ndb-7.5.17) (2020-01-14, General Availability)

MySQL NDB Cluster 7.5.17 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

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)

- 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)
- 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)
- 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 <code>ODirect</code> was enabled for an EOF backup data file write whose size was less than 512 bytes and the backup was in the <code>STOPPING</code> 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 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 7.5.16 (5.7.28-ndb-7.5.16) (2019-10-15, General Availability)

MySQL NDB Cluster 7.5.16 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

- · 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)

Bugs Fixed

• 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)
- 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.

- 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)
- 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)

Changes in MySQL NDB Cluster 7.5.15 (5.7.27-ndb-7.5.15) (2019-07-23, General Availability)

MySQL NDB Cluster 7.5.15 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

- · Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

• Building with CMake3 is now supported by the compile-cluster script included in the NDB source distribution. (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.
- 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)
- 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 resourceconstrained environments. (Bug #29528188)
- 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 7.5.14 (5.7.26-ndb-7.5.14) (2019-04-26, General Availability)

MySQL NDB Cluster 7.5.14 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

Bugs Fixed

- NDB Disk Data: NDB did not validate MaxNoOfOpenFiles in relation to InitialNoOfOpenFiles correctly, leading data nodes to fail with an error message that did not make the nature of the problem clear 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.

- 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)
- 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.

 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".

(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 7.5.13 (5.7.25-ndb-7.5.13) (2019-01-22, General Availability)

MySQL NDB Cluster 7.5.13 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the $_{\rm NDB}$ storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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, General Availability)).

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: When writes on the master—done in such a way that multiple changes affecting BLOB column values belonging to the same primary key were part of the same epoch—were

replicated to the slave, Error 1022 occurred due to constraint violations in the NDB\$BLOB_id_part table. (Bug #28746560)

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

No new LCPs could be started.

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

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

References: See also: Bug #11757421.

- 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)
- Executing SELECT * FROM INFORMATION_SCHEMA.TABLES caused SQL nodes to restart in some cases. (Bug #27613173)
- When tables with BLOB columns were dropped and then re-created with a different number of BLOB columns the event definitions for monitoring table changes could become inconsistent in certain error situations involving communication errors when the expected cleanup of the corresponding events was not performed. In particular, when the new versions of the tables had more BLOB columns than the original tables, some events could be missing. (Bug #27072756)
- When running a cluster with 4 or more data nodes under very high loads, data nodes could sometimes fail with Error 899 Rowid already allocated. (Bug #25960230)
- When starting, a data node copies metadata, while a local checkpoint updates metadata. To avoid any conflict, any ongoing LCP activity is paused while metadata is being copied. An issue arose when a local checkpoint was paused on a given node, and another node that was also restarting checked for a complete LCP on this node; the check actually caused the LCP to be completed before copying of metadata was complete and so ended the pause prematurely. Now in such cases, the LCP completion check waits to complete a paused LCP until copying of metadata is finished and the pause ends as expected, within the LCP in which it began. (Bug #24827685)
- Asynchronous disconnection of mysqld from the cluster caused any subsequent attempt to start an NDB API transaction to fail. If this occurred during a bulk delete operation, the SQL layer called HA::end_bulk_delete(), whose implementation by ha_ndbcluster assumed that a transaction had been started, and could fail if this was not the case. This problem is fixed by checking that the transaction pointer used by this method is set before referencing it. (Bug #20116393)
- 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 7.5.12 (5.7.24-ndb-7.5.12) (2018-10-23, General Availability)

MySQL NDB Cluster 7.5.12 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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

Bugs Fixed

- Packaging: Expected NDB header files were in the devel RPM package instead of libndbclient-devel. (Bug #84580, Bug #26448330)
- MySQL NDB ClusterJ: When a table containing a BLOB or a TEXT field was being queried with ClusterJ for a record that did not exist, an exception ("The method is not valid in current blob state") was thrown. (Bug #28536926)
- MySQL NDB ClusterJ: A NullPointerException was thrown when a full table scan was
 performed with ClusterJ on tables containing either a BLOB or a TEXT field. It was because
 the proper object initializations were omitted, and they have now been added by this fix. (Bug
 #28199372, Bug #91242)
- When the SUMA kernel block receives a SUB_STOP_REQ signal, it executes the signal then replies with SUB_STOP_CONF. (After this response is relayed back to the API, the API is open to send more SUB_STOP_REQ signals.) After sending the SUB_STOP_CONF, SUMA drops the subscription if no subscribers are present, which involves sending multiple DROP_TRIG_IMPL_REQ messages to DBTUP. LocalProxy can handle up to 21 of these requests in parallel; any more than this are queued in the Short Time Queue. When execution of a DROP_TRIG_IMPL_REQ was delayed, there was a chance for the queue to become overloaded, leading to a data node shutdown with Error in short time queue.

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

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

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

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

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

- 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)
- 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 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.

Changes in MySQL NDB Cluster 7.5.11 (5.7.23-ndb-7.5.11) (2018-07-27, General Availability)

MySQL NDB Cluster 7.5.11 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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

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)
- MySQL NDB ClusterJ: ClusterJ could not be built from source using JDK 9. (Bug #27977985)
- An internal buffer being reused immediately after it had been freed could lead to an unplanned data node shutdown. (Bug #27622643)

References: See also: Bug #28698831.

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

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

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

References: See also: Bug #27566346.

• 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.

- 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.

• 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).

• 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 7.5.10 (5.7.22-ndb-7.5.10) (2018-04-20, General Availability)

MySQL NDB Cluster 7.5.10 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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

Bugs Fixed

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

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

- NDB Cluster APIs: Incorrect results, usually an empty result set, were returned when setBound()
 was used to specify a NULL bound. This issue appears to have been caused by a problem in gcc, limited to cases using the old version of this method (which does not employ NdbRecord), and is fixed by rewriting the problematic internal logic in the old implementation. (Bug #89468, Bug #27461752)
- 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)
- 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)
- Under certain conditions, data nodes restarted unnecessarily during execution of ALTER TABLE... REORGANIZE PARTITION. (Bug #25675481)

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

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

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

References: See also: Bug #20112700.

- ndb_restore sometimes logged data file and log file progress values much greater than 100%. (Bug #20989106)
- 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)
- ndb_restore --print-data --hex did not print trailing 0s of LONGVARBINARY values. (Bug #65560, Bug #14198580)

Changes in MySQL NDB Cluster 7.5.9 (5.7.21-ndb-7.5.9) (2018-01-17, General Availability)

MySQL NDB Cluster 7.5.9 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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 5.7.21 (see Changes in MySQL 5.7.21 (2018-01-15, General Availability)).

Bugs Fixed

• NDB Replication: On an SQL node not being used for a replication channel with sql_log_bin=0 it was possible after creating and populating an NDB table for a table map event to be written to the

binary log for the created table with no corresponding row events. This led to problems when this log was later used by a slave cluster replicating from the mysqld where this table was created.

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

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

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

In cases where closing of the file was slow, it was possible for more than 4 files per redo log part to be open concurrently; since these files were opened using the <code>OM_WRITE_BUFFER</code> 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)
- In certain circumstances where multiple Ndb objects were being used in parallel from an API node,
 the block number extracted from a block reference in DBLQH was the same as that of a SUMA block
 even though the request was coming from an API node. Due to this ambiguity, DBLQH mistook the
 request from the API node for a request from a SUMA block and failed. This is fixed by checking node
 IDs before checking block numbers. (Bug #88441, Bug #27130570)
- 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)
- 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 <code>glibc</code> version is used. (Bug #87683, Bug #26758939)

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

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 7.5.8 (5.7.20-ndb-7.5.8) (2017-10-18, General Availability)

MySQL NDB Cluster 7.5.8 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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

- · Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

 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)

Bugs Fixed

- 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)
- 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)

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

Changes in MySQL NDB Cluster 7.5.7 (5.7.19-ndb-7.5.7) (2017-07-19, General Availability)

MySQL NDB Cluster 7.5.7 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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 5.7.19 (see Changes in MySQL 5.7.19 (2017-07-17, General Availability)).

- Packaging Notes
- Functionality Added or Changed
- Bugs Fixed

Packaging Notes

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

Functionality Added or Changed

- Important Change; MySQL NDB ClusterJ: The ClusterJPA plugin for OpenJPA is no longer supported by NDB Cluster, and has been removed from the distribution. (Bug #23563810)
- NDB Replication: Added the --ndb-log-update-minimal option for logging by mysqld. This option causes only primary key values to be written in the before image, and only changed columns in the after image. (Bug #24438868)
- 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)
- 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)
- Added the --diff-default option for ndb_config. This option causes the program to print only those parameters having values that differ from their defaults. (Bug #85831, Bug #25844166)
- Added the --query-all option to ndb_config. This option acts much like the --query option
 except that --query-all (short form: -a) dumps configuration information for all attributes at one
 time. (Bug #60095, Bug #11766869)

Bugs Fixed

- 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)

- 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 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 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.

- NDB Cluster APIs: The implementation method NdbDictionary::NdbTableImpl::getColumn(), used from many places in the NDB API where a column is referenced by name, has been made more efficient. This method used a linear search of an array of columns to find the correct column object, which could be inefficient for tables with many columns, and was detected as a significant use of CPU in customer applications. (Ideally, users should perform name-to-column object mapping, and then use column IDs or objects in method calls, but in practice this is not always done.) A less costly hash index implementation, used previously for the name lookup, is reinstated for tables having relatively many columns. (A linear search continues to be used for tables having fewer columns, where the difference in performance is neglible.) (Bug #24829435)
- MySQL NDB ClusterJ: The JTie and NDB JTie tests were skipped when the unit tests for ClusterJ were being run. (Bug #26088583)
- MySQL NDB ClusterJ: Compilation for the tests for NDB JTie failed. It was due to how null
 references were handled, which has been corrected by this fix. (Bug #26080804)
- Backup .log files contained log entries for one or more extra fragments, due to an issue with
 filtering out changes logged by other nodes in the same node group. This resulted in a larger .log
 file and thus use of more resources than necessary; it could also cause problems when restoring,
 since backups from different nodes could interfere with one another while the log was being applied.
 (Bug #25891014)
- When making the final write to a redo log file, it is expected that the next log file is already opened for writes, but this was not always the case with a slow disk, leading to node failure. Now in such cases NDB waits for the next file to be opened properly before attempting to write to it. (Bug #25806659)
- Data node threads can be bound to a single CPU or a set of CPUs, a set of CPUs being represented internally by NDB as a SparseBitmask. When attempting to lock to a set of CPUs, CPU usage was excessive due to the fact that the routine performing the locks used the mt_thr_config.cpp::do_bind() method, which looks for bits that are set over the entire theoretical range of the SparseBitmask (2³²-2, 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)
- 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.

- Setting NoOfFragmentLogParts such that there were more than 4 redo log parts per local data
 manager led to resource exhaustion and subsequent multiple data node failures. Since this is an
 invalid configuration, a check has been added to detect a configuration with more than 4 redo log
 parts per LDM, and reject it as invalid. (Bug #25333414)
- Execution of an online ALTER TABLE ... REORGANIZE PARTITION statement on an NDB table having a primary key whose length was greater than 80 bytes led to restarting of data nodes, causing the reorganization to fail. (Bug #25152165)
- 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)
- The planned shutdown of an NDB Cluster having more than 10 data nodes was not always performed gracefully. (Bug #20607730)
- Due to a previous issue with unclear separation between the optimize and execute phases when a query involved a GROUP BY, the join-pushable evaluator was not sure whether its optimized query execution plan was in fact pushable. For this reason, such grouped joins were always considered not pushable. It has been determined that the separation issue has been resolved by work already done in MySQL 5.6, and so we now remove this limitation. (Bug #86623, Bug #26239591)
- When deleting all rows from a table immediately followed by DROP TABLE, it was possible that the shrinking of the DBACC hash index was not ready prior to the drop. This shrinking is a per-fragment operation that does not check the state of the table. When a table is dropped, DBACC releases resources, during which the description of the fragment size and page directory is not consistent; this could lead to reads of stale pages, and undefined behavior.

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

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

- The internal function <code>execute_signals()</code> in <code>mt.cpp</code> 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, <code>ndbmtd</code> failed with an error. To keep this from occurring, this function now only reads section pointers that are actually passed to it. (Bug #86354, Bug #26092639)
- The ndb_show_tables program --unqualified option did not work correctly when set to 0
 (false); this should disable the option and so cause fully qualified table and index names to be printed
 in the output. (Bug #86017, Bug #25923164)
- When an NDB table with foreign key constraints is created, its indexes are created first, and then, during foreign key creation, these indexes are loaded into the NDB dictionary cache. When a CREATE TABLE statement failed due to an issue relating to foreign keys, the indexes already in the cache were not invalidated. This meant that any subsequent CREATE TABLE with any indexes having the same names as those in the failed statement produced inconsistent results. Now, in such cases, any indexes named in the failed CREATE TABLE are immediately invalidated from the cache. (Bug #85917, Bug #25882950)
- Attempting to execute ALTER TABLE ... ADD FOREIGN KEY when the key to be added had the name of an existing foreign key on the same table failed with the wrong error message. (Bug #85857, Bug #23068914)
- The node internal scheduler (in mt.cpp) collects statistics about its own progress and any outstanding work it is performing. One such statistic is the number of outstanding send bytes, collected in send_buffer::m_node_total_send_buffer_size. This information may later be used by the send thread scheduler, which uses it as a metric to tune its own send performance versus latency.

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

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

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

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

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

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

The DBSPJ block's FLUSH_AI attribute allows DBTUP to produce two TRANSID_AI results from the same row, one for the client, and one for DBSPJ, which is needed for key lookups on the joined tables. The size of both of these were added to the read length reported by the DBTUP block, which caused the controlling DBLQH block to believe that it had consumed more of the available maximum

batch size than was actually the case, leading to premature termination of the scan batch which could have a negative impact on performance of SPJ scans. To correct this, only the actual read length part of an API request is now reported in such cases. (Bug #85408, Bug #25702850)

- Data node binaries for Solaris 11 built using Oracle Developer Studio 12.5 on SPARC platforms failed with bus errors. (Bug #85390, Bug #25695818)
- 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 7.5.6 (5.7.18-ndb-7.5.6) (2017-04-10, General Availability)

MySQL NDB Cluster 7.5.6 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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

- · Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

• Packaging: Yum repo packages are added for EL5, EL6, EL7, and SLES12.

Apt repo packages are added for Debian 7, Debian 8, Ubuntu 14.04, and Ubuntu 16.04

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.

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

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

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

- CPU usage of the data node's main thread by the DBDIH master block as the end of a local
 checkpoint could approach 100% in certain cases where the database had a very large number of
 fragment replicas. This is fixed by reducing the frequency and range of fragment queue checking
 during an LCP. (Bug #25443080)
- The ndb_print_backup_file utility failed when attempting to read from a backup file when the backup included a table having more than 500 columns. (Bug #25302901)

References: See also: Bug #25182956.

- CMake now detects whether a GCC 5.3.0 loop optimization bug occurs and attempts a workaround if so. (Bug #25253540)
- 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.

- 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)
- 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)

• Ubuntu 14.04 and Ubuntu 16.04 are now supported.

Changes in MySQL NDB Cluster 7.5.5 (5.7.17-ndb-7.5.5) (2017-01-17, General Availability)

MySQL NDB Cluster 7.5.5 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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

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)
- **Packaging:** The RPM installer for the MySQL NDB Cluster auto-installer package had a dependency on python2-crypt instead of python-crypt. (Bug #24924607)
- Microsoft Windows: Installation failed when the Auto-Installer (ndb_setup.py) was run on a Windows host that used Swedish as the system language. This was due to system messages being issued using the cp1252 character set; when these messages contained characters that did not map directly to 7-bit ASCII (such as the ä character in Tjänsten ... startar), conversion to UTF-8—as expected by the Auto-Installer web client—failed.

This fix has been tested only with Swedish as the system language, but should work for Windows systems set to other European languages that use the cp1252 character set. (Bug #83870, Bug #25111830)

 ndbinfo Information Database: A number of inconsistencies and other issues had arisen regarding ndbinfo tables due to manual copying of the table and view definitions in the sources. Now the SQL statements for these are generated, for consistency. (Bug #23305078)

References: See also: Bug #25047951.

• 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)
- CMake now avoids configuring the -fexpensive-optimizations option for GCC versions for which the option triggers faulty shift-or optimizations. (Bug #24947597, Bug #83517)
- The rand() function was used to produce a unique table ID and table version needed to identify a schema operation distributed between multiple SQL nodes, relying on the assumption that rand() would never produce the same numbers on two different instances of mysqld. It was later determined that this is not the case, and that in fact it is very likely for the same random numbers to be produced on all SQL nodes.

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

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

- The ndb_show_tables utility did not display type information for hash maps or fully replicated triggers. (Bug #24383742)
- Long message buffer exhaustion when firing immediate triggers could result in row ID leaks; this could later result in persistent RowId already allocated errors (NDB Error 899). (Bug #23723110)

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

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

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

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

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

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

- The NDB binlog injector thread used an injector mutex to perform two important tasks:
 - 1. Protect against client threads creating or dropping events whenever the injector thread waited for pollEvents().
 - 2. Maintain access to data shared by the injector thread with client threads.

The first of these could hold the mutex for long periods of time (on the order of 10ms), while locking it again extremely quickly. This could keep it from obtaining the lock for data access ("starved") for unnecessarily great lengths of time.

To address these problems, the injector mutex has been refactored into two—one to handle each of the two tasks just listed.

It was also found that initialization of the binlog injector thread held the injector mutex in several places unnecessarily, when only local thread data was being initialized and sent signals with condition information when nothing being waited for was updated. These unneeded actions have been removed, along with numerous previous temporary fixes for related injector mutex starvation issues. (Bug #83676, Bug #83127, Bug #25042101, Bug #24715897)

References: See also: Bug #82680, Bug #20957068, Bug #24496910.

• 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)

• When ndbmtd was built on Solaris/SPARC with version 5.3 of the GNU tools, data nodes using the resulting binary failed during startup. (Bug #83500, Bug #24941880)

References: See also: Bug #83517, Bug #24947597.

- MySQL NDB Cluster failed to compile using GCC 6. (Bug #83308, Bug #24822203)
- 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.

Changes in MySQL NDB Cluster 7.5.4 (5.7.16-ndb-7.5.4) (2016-10-18, General Availability)

MySQL NDB Cluster 7.5.4 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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 5.7.16 (see Changes in MySQL 5.7.16 (2016-10-12, General Availability)).

- · Functionality Added or Changed
- · Bugs Fixed

Functionality Added or Changed

• Important Change; Packaging: Naming and organization of the RPMs provided for MySQL NDB Cluster have been changed to align with those released for the MySQL server. All MySQL NDB Cluster RPM package names are now prefixed with mysql-cluster. Data nodes are now installed using the data-node package; management nodes are now installed from the management-server package; and SQL nodes require the server and common packages. Important: SQL nodes must use the mysql-cluster version of these RPMs; the versions released for the standard MySQL server do not provide support for the NDB storage engine. All client programs, including both the mysql client and the ndb_mgm management client, are now included in the client RPM.

For more information, see Installing NDB Cluster from RPM. (WL #9129)

• Important Change: Added the PARTITION_BALANCE values FOR_RA_BY_LDM_X_2, FOR_RA_BY_LDM_X_3, and FOR_RA_BY_LDM_X_4, which can be used to set the number of partitions used by each local data manager to two, three, and four partitions, respectively, in addition to FOR_RA_BY_LDM, which sets this number to one.

Use of $\mathtt{MAX_ROWS}$ for setting the number of partitions used by \mathtt{NDB} tables is now deprecated and subject to removal in a future MySQL NDB Cluster version.

For more information, see Setting NDB Comment Options. (Bug #81759, Bug #23544301)

• MySQL NDB ClusterJ: To help applications handle database errors better, a number of new features have been added to the ClusterJDatastoreException class:

- A new method, getCode(), returns code from the NdbError object.
- A new method, getMysqlCode(), returns mysql_code from the NdbError object.
- A new subclass, ClusterJDatastoreException.Classification, gives users the ability to decode the result from getClassification(). The method Classification.toString() gives the name of the error classification as listed in NDB Error Classifications.

(Bug #22353594)

• Added the --print-sql-log option for the ndb_restore program included with the MySQL NDB Cluster distribution. This option causes the program to log SQL statements to stdout.

Note that each table being restored in this fashion must have an explicitly defined primary key; the hidden primary key implemented by the NDB storage engine is *not* sufficient for this purpose. (Bug #13511949)

- For fully replicated tables, ndb_desc shows only nodes holding main fragment replicas for partitions; nodes with copy fragment replicas only are ignored. To make this information available in the mysql client, several new tables have been introduced in the ndbinfo information database. These tables are listed here, with brief descriptions:
 - dict_obj_info provides the names and types of database (DICT) objects in NDB, such as tables and indexes, as well as information about parent objects where applicable
 - table_distribution_status provides NDB table distribution status information
 - table_fragments provides information about the distribution of NDB table fragments
 - table_info provides information about logging, checkpointing, storage, and other options in force for each NDB table
 - table_replicas provides information about fragment replicas.

For more information, see ndbinfo: The NDB Cluster Information Database. (Bug #81762, Bug #23547643)

Bugs Fixed

- Important Change: The default value of the --ndb-default-column-format server option has been changed from DYNAMIC to FIXED. This has been done for backwards compatibility. Only the default has been changed; setting this option to DYNAMIC continues to cause DYNAMIC to be used for ROW_FORMAT and COLUMN_FORMAT unless overridden. (Bug #24487363)
- Important Change: Event buffer status reporting has been improved by altering the semantics for calculating lag or slippage. Rather than defining this lag as the number of epochs behind, lag is now taken as the number of epochs completely buffered in the event buffer, but not yet consumed by the binlog injector thread. As part of this work, the default value for the ndb_report_thresh_binlog_epoch_slip system variable has been increased from 3 to 10. For more information, see the description of this variable in the documentation, as well as Event Buffer Reporting in the Cluster Log. (Bug #22916457)

References: See also: Bug #22901309.

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

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

NDB Cluster APIs: To process incoming signals, a thread which wants to act as a receiver must
acquire polling rights from the transporter layer. This can be requested and assigned to a separate
receiver thread, or each client thread can take the receiver role when it is waiting for a result.

When the thread acting as poll owner receives a sufficient amount of data, it releases locks on any other clients taken while delivering signals to them. This could make them runnable again, and the operating system scheduler could decide that it was time to wake them up, which happened at the expense of the poll owner threads, which were in turn excluded from the CPU while still holding polling rights on it. After this fix, polling rights are released by a thread before unlocking and signalling other threads. This makes polling rights available for other threads that are actively executing on this CPU.

This change increases concurrency when polling receiver data, which should also reduce latency for clients waiting to be woken up. (Bug #83129, Bug #24716756)

• NDB Cluster APIs: libndbclient and libmysqlclient exported conflicting symbols, resulting in a segmentation fault in debug builds on Linux. To fix this issue, the conflicting symbols in libndbclient.so are no longer publicly visible. Due to this change, the version number for libndbclient.so has been raised from 6.0.0 to 6.1.0. (Bug #83093, Bug #24707521)

References: See also: Bug #80352, Bug #22722555.

• NDB Cluster APIs: When NDB schema object ownership checks are enabled by a given NdbTransaction, objects used by this transaction are checked to make sure that they belong to the NdbDictionary owned by this connection. An attempt to create a NdbOperation, NdbScanOperation, or NdbIndexScanOperation on a table or index not belonging to the same connection fails.

This fix corrects a resource leak which occurred when the operation object to be created was allocated before checking schema object ownership and subsequently not released when the object creation failed. (Bug #81949, Bug #23623978)

References: See also: Bug #81945, Bug #23623251.

• NDB Cluster APIs: NDB API objects are allocated in the context of an Ndb object, or of an NdbTransaction object which is itself owned by an Ndb object. When a given Ndb object is destroyed, all remaining NdbTransaction objects are terminated, and all NDB API objects related to this Ndb object should be released at this time as well. It was found, when there remained unclosed NdbTransaction objects when their parent Ndb object was destroyed, leaks of objects allocated from the NdbTransaction objects could occur. (However, the NdbTransaction objects themselves did not leak.)

While it is advisable (and, indeed, recommended) to close an NdbTransaction explicitly as soon as its lifetime ends, the destruction of the parent Ndb object should be sufficient to release whatever objects are dependent on it. Now in cases such as described previously, the Ndb destructor checks to ensure that all objects derived from a given Ndb instance are truly released. (Bug #81945, Bug #23623251)

• NDB Cluster APIs: The term "fragment count type" has been superceded by "partition balance". This change affects NDB_TABLE options for NDB tables as well as in the NDB API. In NDB_TABLE table option syntax, the FRAGMENT_COUNT_TYPE keyword is replaced with PARTITION_BALANCE. In the NDB API, the Table methods getFragmentCountType() and setFragmentCountType() have been renamed to getPartitionBalance() and setPartitionBalance(), respectively; getFragmentCountTypeString() is renamed to

getPartitionBalanceString(). In addition, Object::FragmentCountType has been renamed to PartitionBalance, and the names of its enumerated values have been updated to be consistent with the new nomenclature.

For more information on how these changes affect NDB API applications, see the indicated Table and Object member descriptions. For more information on the SQL-level changes made as part of this fix, Setting NDB Comment Options. (Bug #81761, Bug #23547525)

References: See also: Bug #83147, Bug #24733331.

• NDB Cluster APIs: In some of the NDB API example programs included with the MySQL NDB Cluster distribution, ndb_end() was called prior to calling the Ndb_cluster_connection destructor. This caused a segmentation fault in debug builds on all platforms. The example programs affected have also been extensively revised and refactored. See NDB API Examples, for more information. (Bug #80352, Bug #22722555)

References: See also: Bug #83093, Bug #24707521.

- If more than 4096 seconds elapsed while calculating an internal NdbDuration::microSec() value, this could cause an assert warning that the calculation would overflow. We fix this to avoid any overflow or precision loss when converting from the internal "tick" format to microseconds and nanoseconds, by performing the calculation in two parts corresponding to seconds and fractions of a second. (Bug #24695026)
- The serial commit protocol—which commits each operation at each replica individually and serially, and is used by the DBTC kernel block (see The DBTC Block) for takeover and when a transaction is judged to have timed out during the COMMIT or COMPLETE phase—had no support for LATE_COMMIT, which is required for the READ_BACKUP and FULLY_REPLICATED protocols. (Bug #24681305)
- In some cases, ALTER TABLE ... REORGANIZE PARTITION could lead to an unplanned shutdown of the cluster. This was due to the fact that, for fully replicated tables. the log part ID was assumed to be the same as the partition ID. This worked when FOR_RA_BY_LDM was used, but not necessarily for the other partition balancing types. (Bug #24610551)
- Using ALGORITHM=INPLACE when changing any of a table's NDB_TABLE properties (see Setting NDB Comment Options) caused the server to fail. (Bug #24584741)
- Following successive ALTER TABLE statements updating NDB_TABLE properties (see Setting
 NDB Comment Options), the current values were not always shown by SHOW CREATE TABLE or
 ndb_desc. (Bug #24584690)
- Case-insensitivity of keywords such as FULLY_REPLICATED in NDB_TABLE comments was not honored. (Bug #24577931)
- An ALTER TABLE statement attempting to set both FULLY_REPLICATED and PARTITION_BALANCE (see Setting NDB Comment Options) failed with a garbled error message. (Bug #24577894)
- A number of dependencies between the binlog injector thread and the NDB utility thread—a recurring source of synchronization and other problems—were removed. The principal changes are listed here:
 - Moved the setup of binlog injector structures from the utility thread to the injector thread itself.
 - Removed sharing of some utility and injector thread structures between these threads.
 - Moved stopping of the utility thread from the injector thread into a common block in which other such threads are stopped.
 - Removed a number of hacks required by the previous design.

 Removed some injector mutex locking and injector condition signaling which were made obsolete by the changes already listed.

(Bug #24496910)

References: See also: Bug #22204186.

 A late commit ACK signal used for FULLY_REPLICATED or READ_BACKUP tables caused the associated ApiConnectionRecord to have an invalid state. (Bug #24459817)

References: See also: Bug #24444861.

- Added missing error information for a failure occurring when tables on disk became full. (Bug #24425373)
- When ndbmtd crashed, the resulting error log incorrectly specified the name of the trace for thread
 0, appending the nonexistent suffix _t0 to the file name. (Bug #24353408)
- Passing a nonexistent node ID to CREATE NODEGROUP led to random data node failures. (Bug #23748958)
- DROP TABLE followed by a node shutdown and subesequent master takeover—and with the containing local checkpoint not yet complete prior to the takeover—caused the LCP to be ignored, and in some cases, the data node to fail. (Bug #23735996)

References: See also: Bug #23288252.

- Removed an invalid assertion to the effect that all cascading child scans are closed at the time
 API connection records are released following an abort of the main transaction. The assertion was
 invalid because closing of scans in such cases is by design asynchronous with respect to the main
 transaction, which means that subscans may well take some time to close after the main transaction
 is closed. (Bug #23709284)
- Although arguments to the DUMP command are 32-bit integers, ndb_mgmd used a buffer of only 10 bytes when processing them. (Bug #23708039)
- The READ_BACKUP setting was not honored when performing scans on BLOB tables. (Bug #23703536)
- Setting FULLY_REPLICATED=1 (see Setting NDB Comment Options) did not propagate to the internal BLOB part tables used for BLOB and TEXT columns. (Bug #23703343)
- The READ_BACKUP setting was not applied to unique indexes. (Bug #23702848)
- In ReadCommitted mode, DBSPJ read primary fragment replicas for tables with READ_BACKUP (see Setting NDB Comment Options), even when a local fragment was available. (Bug #23633848)
- ALL REPORT MemoryUsage produced incorrect output when fully replicated tables were in use. (Bug #23539805)
- Ordered indexes did not inherit READ_BACKUP (see Setting NDB Comment Options) from an indexed table, which meant that ordered index scans continued to be routed to only to primary fragment replicas and never to backup fragment replicas.

Now DBDICT sets this property on ordered indexes from the table property when it distributes this information to instances of DBTC and DBSPJ. (Bug #23522027)

- Updates to a table containing a virtual column could cause the binary logging thread to fail. (Bug #23514050)
- A number of potential buffer overflow issues were found and fixed in the NDB codebase. (Bug #23152979)

- During an online upgrade from a MySQL NDB Cluster 7.3 release to an NDB 7.4 (or later) release, the failures of several data nodes running the lower version during local checkpoints (LCPs), and just prior to upgrading these nodes, led to additional node failures following the upgrade. This was due to lingering elements of the EMPTY_LCP protocol initiated by the older nodes as part of an LCP-plus-restart sequence, and which is no longer used in NDB 7.4 and later due to LCP optimizations implemented in those versions. (Bug #23129433)
- A SIGNAL_DROPPED_REP handler invoked in response to long message buffer exhaustion was defined in the SPJ kernel block, but not actually used. This meant that the default handler from SimulatedBlock was used instead in such cases, which shut down the data node. (Bug #23048816)

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

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

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

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

- ndb_restore set the MAX_ROWS attribute for a table for which it had not been set prior to taking the backup. (Bug #22904640)
- Whenever data nodes are added to or dropped from the cluster, the NDB kernel's Event API is notified of this using a SUB_GCP_COMPLETE_REP signal with either the ADD (add) flag or SUB (drop) flag set, as well as the number of nodes to add or drop; this allows NDB to maintain a correct count of SUB_GCP_COMPLETE_REP signals pending for every incomplete bucket. In addition to handling the bucket for the epoch associated with the addition or removal, it must also compensate for any later incomplete buckets associated with later epochs. Although it was possible to complete such buckets out of order, there was no handling of these, leading a stall in to event reception.

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

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

- When performing online reorganization of tables, unique indexes were not included in the reorganization. (Bug #13714258)
- Under very high loads, neither the receive thread nor any user thread had sufficient capacity to
 handle poll ownership properly. This meant that, as the load and the number of active thread
 increased, it became more difficult to sustain throughput. This fix attempts to increase the priority of
 the receive thread and retains poll ownership if successful.

This fix requires sufficient permissions to be enabled. On Linux systems, this means ensuring that either the data node binary or the user it runs as has permission to change the nice level. (Bug #83217, Bug #24761073)

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

- Local reads of unique index and blob tables did not work correctly for fully replicated tables using more than one node group. (Bug #83016, Bug #24675602)
- The effects of an ALTER TABLE statement changing a table to use READ_BACKUP were not preserved after a restart of the cluster. (Bug #82812, Bug #24570439)
- Using FOR_RP_BY_NODE or FOR_RP_BY_LDM for PARTITION_BALANCE did not work with fully replicated tables. (Bug #82801, Bug #24565265)
- Changes to READ_BACKUP settings were not propagated to internal blob tables. (Bug #82788, Bug #24558232)
- The count displayed by the c_exec column in the ndbinfo.threadstat table was incomplete. (Bug #82635, Bug #24482218)
- The default PARTITION_BALANCE setting for NDB tables created with READ_BACKUP=1 (see Setting NDB Comment Options) has been changed from FOR_RA_BY_LDM to FOR_RP_BY_LDM. (Bug #82634, Bug #24482114)
- The internal function ndbcluster_binlog_wait(), which provides a way to make sure that all events originating from a given thread arrive in the binary log, is used by SHOW BINLOG EVENTS as well as when resetting the binary log. This function waits on an injector condition while the latest global epoch handled by NDB is more recent than the epoch last committed in this session, which implies that this condition must be signalled whenever the binary log thread completes and updates a new latest global epoch. Inspection of the code revealed that this condition signalling was missing, and that, instead of being awakened whenever a new latest global epoch completes (~100ms), client threads waited for the maximum timeout (1 second).

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

- Fully replicated internal foreign key or unique index triggers could fire multiple times, which led to aborted transactions for an insert or a delete operation. This happened due to redundant deferred constraint triggers firing during pre-commit. Now in such cases, we ensure that only triggers specific to unique indexes are fired in this stage. (Bug #82570, Bug #24454378)
- Backups potentially could fail when using fully replicated tables due to their high usage (and subsequent exhaustion) of internal trigger resources. To compensate for this, the amount of memory reserved in the NDB kernel for internal triggers has been increased, and is now based in part on the maximum number of tables. (Bug #82569, Bug #24454262)

References: See also: Bug #23539733.

In the DBTC function executeFullyReplicatedTrigger() in the NDB kernel, an incorrect check
of state led in some cases to failure handling when no failure had actually occurred. (Bug #82568,
Bug #24454093)

References: See also: Bug #23539733.

When returning from LQHKEYREQ with failure in LQHKEYREF in an internal trigger operation, no
check was made as to whether the trigger was fully replicated, so that those triggers that were fully
replicated were never handled. (Bug #82566, Bug #24453949)

References: See also: Bug #23539733.

- When READ_BACKUP had not previously been set, then was set to 1 as part of an ALTER
 TABLE ... ALGORITHM=INPLACE statement, the change was not propagated to internal unique
 index tables or BLOB tables. (Bug #82491, Bug #24424459)
- Distribution of MySQL privileges was incomplete due to the failure of the mysql_cluster_move_privileges() procedure to convert the mysql.proxies_priv table to

NDB. The root cause of this was an ALTER TABLE ... ENGINE NDB statement which sometimes failed when this table contained illegal TIMESTAMP values. (Bug #82464, Bug #24430209)

- The internal variable m_max_warning_level was not initialized in storage/ndb/src/kernel/blocks/thrman.cpp. This sometimes led to node failures during a restart when the uninitialized value was treated as 0. (Bug #82053, Bug #23717703)
- Usually, when performing a system restart, all nodes are restored from redo logs and local checkpoints (LCPs), but in some cases some node might require a copy phase before it is finished with the system restart. When this happens, the node in question waits for all other nodes to start up completely before performing the copy phase. Notwithstanding the fact that it is thus possible to begin a local checkpoint before reaching start phase 4 in the DBDIH block, LCP status was initialized to IDLE in all cases, which could lead to a node failure. Now, when performing this variant of a system restart, the LCP status is no longer initialized. (Bug #82050, Bug #23717479)
- After adding a new node group online and executing ALTER TABLE ... ALGORITHM=INPLACE REORGANIZE PARTITION, partition IDs were not set correctly for new fragments.

In a related change done as part of fixing this issue, ndb_desc -p now displays rows relating to partitions in order of partition ID. (Bug #82037, Bug #23710999)

- When executing STOP BACKUP it is possible sometimes that a few bytes are written to the backup
 data file before the backup process actually terminates. When using ODIRECT, this resulted in the
 wrong error code being returned. Now in such cases, nothing is written to O_DIRECT files unless the
 alignment is correct. (Bug #82017, Bug #23701911)
- When transaction coordinator (TC) connection records were used up, it was possible to handle scans only for local checkpoints and backups, so that operations coming from the DBUTIL block—used for ALTER TABLE ... REORGANIZE PARTITION and other operations that reorganize metadata—were unnecessarily blocked. In addition, such operations were not always retried when TC records were exhausted. To fix this issue, a number of operation records are now earmarked for DBUTIL usage, as well as for LCP and backup usage so that these operations are also not negatively impacted by operations coming from DBUTIL.

For more information, see The DBUTIL Block. (Bug #81992, Bug #23642198)

- Operations performing multiple updates of the same row within the same transaction could sometimes lead to corruption of lengths of page entries. (Bug #81938, Bug #23619031)
- During a node restart, a fragment can be restored using information obtained from local checkpoints (LCPs); up to 2 restorable LCPs are retained at any given time. When an LCP is reported to the DIH kernel block as completed, but the node fails before the last global checkpoint index written into this LCP has actually completed, the latest LCP is not restorable. Although it should be possible to use the older LCP, it was instead assumed that no LCP existed for the fragment, which slowed the restart process. Now in such cases, the older, restorable LCP is used, which should help decrease long node restart times. (Bug #81894, Bug #23602217)
- Optimized node selection (ndb_optimized_node_selection setting) was not respected by ndb_data_node_neighbour when this was enabled. (Bug #81778, Bug #23555834)
- NDB no longer retries a global schema lock if this has failed due to a timeout (default 3000ms) and there is the potential for this lock request to participate in a metadata lock-global schema lock deadlock. Now in such cases it selects itself as a "victim", and returns the decision to the requestor of the metadata lock, which then handles the request as a failed lock request (preferable to remaining deadlocked indefinitely), or, where a deadlock handler exists, retries the metadata lock-global schema lock. (Bug #81775, Bug #23553267)

- Two issues were found in the implementation of hash maps—used by NDB for mapping a table row's hash value to a partition—for fully replicated tables:
 - Hash maps were selected based on the number of fragments rather than the number of partitions. This was previously undetected due to the fact that, for other kinds of tables, these values are always the same.
 - 2. The hash map was employed as a partition-to-partition map, using the table row's hash value modulus the partition count as input.

This fix addresses both of the problems just described. (Bug #81757, Bug #23544220)

References: See also: Bug #81761, Bug #23547525, Bug #23553996.

Using mysqld together with --initialize and --ndbcluster led to problems later when
attempting to use mysql_upgrade. When running with --initialize, the server does not require
NDB support, and having it enabled can lead to issues with ndbinfo tables. To prevent this from
happening, using the --initialize option now causes mysqld to ignore the --ndbcluster
option if the latter is also specified.

This issue affects upgrades from MySQL NDB Cluster 7.5.2 or 7.5.3 only. In cases where such upgrades fail for the reasons outlined previously, you can work around the issue by deleting all .frm files in the data/ndbinfo directory following a rolling restart of the entire cluster, then running mysql_upgrade. (Bug #81689, Bug #23518923)

References: See also: Bug #82724, Bug #24521927.

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

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

- ALTER TABLE ... ALGORITHM=INPLACE on a fully replicated table did not copy the associated trigger ID, leading to a failure in the DBDICT kernel block. (Bug #81544, Bug #23330359)
- The list of deferred tree node lookup requests created when preparing to abort a DBSPJ request were not cleared when this was complete, which could lead to deferred operations being started even after the DBSPJ request aborted. (Bug #81355, Bug #23251423)

References: See also: Bug #23048816.

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

References: See also: Bug #23048816.

• The read backup feature added in MySQL NDB Cluster 7.5.2 that makes it possible to read from backup fragment replicas was not used for reads with lock, or for reads of BLOB tables or unique key tables where locks were upgraded to reads with lock. Now the TCKEYREQ and SCAN_TABREQ signals use a flag to convey information about such locks making it possible to read from a backup fragment replica when a read lock was upgraded due to being the read of the base table for a BLOB table, or due to being the read for a unique key. (Bug #80861, Bug #23001841)

 Primary fragment replicas of partitioned tables were not distributed evenly among node groups and local data managers.

As part of the fix for this issue, the maximum number of node groups supported for a single MySQL NDB Cluster, which was previously not determined, is now set at 48 (MAX_NDB_NODE_GROUPS). (Bug #80845, Bug #22996305)

- Several object constructors and similar functions in the NDB codebase did not always perform sanity checks when creating new instances. These checks are now performed under such circumstances. (Bug #77408, Bug #21286722)
- An internal call to malloc() was not checked for NULL. The function call was replaced with a direct write. (Bug #77375, Bug #21271194)

Changes in MySQL NDB Cluster 7.5.3 (5.7.13-ndb-7.5.3) (2016-07-08, Release Candidate)

MySQL NDB Cluster 7.5.3 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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 5.7.13 (see Changes in MySQL 5.7.13 (2016-06-02, General Availability)).

- · Functionality Added or Changed
- · Bugs Fixed

Functionality Added or Changed

• Important Change: It is now possible to set READ_BACKUP for an existing table online using an SQL statement such as ALTER TABLE . . . ALGORITHM=INPLACE, COMMENT="NDB_TABLE=READ_BACKUP=1". See Setting NDB Comment Options, for further information about the READ_BACKUP option. (Bug #80858, Bug #23001617)

References: See also: Bug #18435416.

- Added three new tables to the ndbinfo information database to provide running information about locks and lock attempts in an active MySQL NDB Cluster. These tables, with brief descriptions, are listed here:
 - cluster_locks: Current lock requests which are waiting for or holding locks; this information can be useful when investigating stalls and deadlocks. Analogous to cluster_operations.
 - locks_per_fragment: Counts of lock claim requests, and their outcomes per fragment, as well as total time spent waiting for locks successfully and unsuccessfully. Analogous to operations_per_fragment and memory_per_fragment.
 - server_locks: Subset of cluster transactions—those running on the local mysqld, showing a connection ID per transaction. Analogous to server_operations.

For more information, see ndbinfo: The NDB Cluster Information Database. (WL #7656)

 The NDB storage engine now supports generated columns (see CREATE TABLE and Generated Columns) as well as indexes on stored generated columns. Indexes on virtual generated columns of NDB tables are not supported.

ALTER TABLE statements that add stored generated columns to NDB tables cannot be performed online. (WL #8188, WL #9048)

Bugs Fixed

- NDB Cluster APIs: The scan lock takeover issued by NdbScanOperation::lockCurrentTuple() did not set the operation type for the takeover operation. (Bug #23314028)
- The ndbinfo cpustat_1sec and cpustat_20sec tables did not provide any history information. (Bug #23520271)
- During shutdown, the mysqld process could sometimes hang after logging NDB Util: Stop ... NDB Util: Wakeup. (Bug #23343739)

References: See also: Bug #21098142.

 During expansion or reduction of a hash table, allocating a new overflow page in the DBACC kernel block caused the data node to fail when it was out of index memory. This could sometimes occur after a large table had increased or decreased very rapidly in size. (Bug #23304519)

References: This issue is a regression of: Bug #13436216.

- When tracking time elapsed from sending a SCAN_FRAGREQ signal to receiving a corresponding SCAN_FRAGCONF, the assumption was made in the DBTC kernel block that a SCAN_FRAGCONF can occur only after sending a SCAN_FRAGREQ or SCAN_NEXTREQ signal, which is not always the case: It is actually possible that a local query handler can, immediately after sending a SCAN_FRAGCONF, send an additional SCAN_FRAGCONF signal upon reporting that the scan is closed. This problem is fixed by ensuring that the timer value is initialized each time before use. (Bug #81907, Bug #23605817, Bug #23306695)
- Table indexes were listed in the output of ndb_desc in a nondeterministic order that could vary between platforms. Now these indexes are ordered by ID in the output. (Bug #81763, Bug #23547742)
- Following a restart of the cluster, the first attempt to read from any of the ndbinfo cpustat, cpustat_50ms, cpustat_1sec, or cpustat_20sec tables generated a warning to the effect that columns were missing from the table. Subsequently, the thread_sleeping and spin_time columns were found to be missing from each of these tables. (Bug #81681, Bug #23514557)

References: See also: Bug #23305078.

• Using a ThreadConfig parameter value with a trailing comma led to an assertion. (Bug #81588, Bug #23344374)

Changes in MySQL NDB Cluster 7.5.2 (5.7.12-ndb-7.5.2) (2016-06-01, Development Milestone)

MySQL NDB Cluster 7.5.2 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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 5.7.12 (see Changes in MySQL 5.7.12 (2016-04-11, General Availability)).

- · Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

• Important Change; NDB Replication: The ndb_binlog_index table now uses the InnoDB storage engine. When upgrading previous versions to the current release or to a later one, use mysql_upgrade with --force --upgrade-system-tables to have it perform ALTER TABLE ... ENGINE=INNODB on this table. Use of the MyISAM storage engine for this table continues to be supported for backward compatibility.

One benefit of this change is that it is now possible to depend on transactional behavior and lock-free reads for this table, which should help alleviate concurrency issues during purge operations and log rotation, and improve the availability of this table.

Due to differences in storage requirements for MyISAM and InnoDB tables, users should be aware that ndb binlog index may now take up more disk space than was previously required.

For more information, see NDB Cluster Replication Schema and Tables. (WL #7162)

• **Performance:** A deficiency in event buffer memory allocation was identified as inefficient and possibly leading to undesirable results. This could happen when additional memory was allocated from the operating system to buffer received event data even when memory had already been allocated but remained unused. This is fixed by allocating the event buffer memory directly from the page allocation memory manager (mmap()), where such functionality is offered by the operating system, allowing for direct control over the memory such that it is in fact returned to the system when released.

This remimplementation avoids the tendencies of the existing one to approach worst-case memory usage, maintainence of data structures for a worst-case event buffer event count, and useless caching of free memory in unusable positions. This work should also help minimize the runtime costs of buffering events, minimize heap fragmentation, and avoid OS-specific problems due to excessive numbers of distinct memory mappings.

In addition, the relationship between epochs and internal EventData objects is now preserved throughout the event lifecycle, reception to consumption, thus removing the need for iterating, and keeping in synch, two different lists representing the epochs and their EventData objects.

As part of this work, better reporting on the relevant event buffer metrics is now provided in the cluster logs. (WL #7677, WL #9207)

References: See also: Bug #21651536, Bug #21660947, Bug #21661297, Bug #21673318, Bug #21689380, Bug #21809959.

- NDB Cluster APIs: Added the Ndb::setEventBufferQueueEmptyEpoch() method, which makes it possible to enable queuing of empty events (event type TE_EMPTY). (Bug #22157845)
- NDB Cluster APIs: Ndb_free_list_t is a template used in the implementation of the NDB API to create a list keeping released API objects such as NdbTransaction, NdbOperation, and NdbRecAttr. One drawback to this template is that released API objects are kept in the list for the lifetime of the owning Ndb object, such that a transient peak in the demand for any object causes an effective leak in memory that persists until this Ndb object itself has been released.

This work adds statistics to each Ndb_free_list instance which samples the usage of objects maintained by the list; now, when objects are released, they can be released into the free list, or deallocated, based on the collected usage statistics. (WL #8351)

- **JSON:** The NDB storage engine supports the MySQL JSON data type and MySQL JSON functions implemented in MySQL 5.7.8 and later. This support is subject to the limitation that a single NDB table can have at most 3 JSON columns. (WL #9007)
- MySQL NDB ClusterJ: To make it easier for ClusterJ to handle fatal errors that require the
 SessionFactory to be closed, a new public method in the SessionFactory interface,
 getConnectionPoolSessionCounts(), has been created. When it returns zeros for all pooled
 connections, it means all sessions have been closed, at which point the SessionFactory can be
 closed and reopened. See Error Handling and Reconnection for more detail. (Bug #22353594)
- Made the following enhancements and additions to the ThreadConfig multithreaded data node (ndbmtd) configuration parameter:
 - Added support for non-exclusive CPU locking on FreeBSD and Windows using cpubind and cpuset.
 - Added support for exclusive CPU locking on Solaris, using cpubind_exclusive and cpuset_exclusive, which are added in this release.
 - Added thread prioritization using thread_prio, which is added in this release. thread_prio is supported on Linux, FreeBSD, Windows, and Solaris, but the exact effects of this setting are platform-specific; see the documentation for details.
 - Added support for setting realtime on Windows platforms.

For more information, see the description of the ThreadConfig parameter in the online documentation. (Bug #25830247, WL #9096)

- ndb_restore now performs output logging for specific stages of its operation. (Bug #21097957)
- An improvement in the hash index implementation used by MySQL NDB Cluster data nodes means that partitions may now contain more than 16 GB of data for fixed columns, and the maximum partition size for fixed column data is now increased to 128 TB. The previous limitation originated with the DBACC block in the NDB kernel using only 32-bit references to the fixed-size part of a row handled in the DBTUP block, even though 45-bit references were already in use elsewhere in the kernel outside the DBACC block; all such references in DBACC now use 45-bit pointers instead.

As part of this work, error messages returned by the DBACC kernel block that were overly generic have now been improved by making them more specific. (Bug #13844581, Bug #17465232, WL #8570, WL #8817, WL #8962)

- A number of changes and improvements were made to the handling of send threads by NDB. The changes are summarized with brief descriptions in the following list:
 - Decreased resource requirements for send threads, making sure that a given configuration using send threads outperforms the same configuration without send threads.
 - Made use of otherwise idle threads (other than receiver threads) as send threads without incurring extra CPU resources in low-load situations.
 - · Improved response time for write transactions.
 - Provided for handling of bad configuration data in a more graceful manner.
 - Made it possible to measure CPU usage with improved real-time reporting from data nodes on their resource usage.

As part of implementing the last item of those just listed, a number of new tables providing information about CPU and thread activity by node, thread ID, and thread type have been added to the ndbinfo information database. These tables are listed here:

- cpustat: Provides per-second, per-thread CPU statistics
- cpustat_50ms: Shows raw per-thread CPU statistics data, gathered every 50ms
- cpustat_1sec: Provides raw per-thread CPU statistics data, gathered each second
- cpustat_20sec: Displays raw per-thread CPU statistics data, gathered every 20 seconds
- threads: Shows names and descriptions of thread types

For more information, see ndbinfo: The NDB Cluster Information Database. (WL #8968)

- Rewrote the implementation of the NDB storage engine's global schema lock functionality to make use of the metadata lock hooks implemented in the MySQL Server in MySQL 5.7.11. (WL #8331)
- A number of improvements provide additional read scalability for NDB by making it possible to
 read tables locally. It is now possible to enable reads from any fragment replica, rather than from
 the primary fragment replica only. This is disabled by default to remain compatible with previous
 behavior, but can be enabled for a given SQL node using the ndb_read_backup system variable
 added in this release.

It also becomes possible to be more flexible about the assignment of partitions by setting a fragment count type. Possible count types are one per node, one per node group, one per Local Data Manager (LDM) per node (this is the previous assignment scheme and still the default), and one per LDM per node group. This setting can be controlled for individual tables by means of a FRAGMENT_COUNT_TYPE option embedded in an NDB_TABLE comment in CREATE TABLE or ALTER TABLE.

This also means that, when restoring table schemas, ndb_restore --restore-meta now uses the default partitioning for the target cluster, rather than duplicating the partitioning of the cluster from which the backup was taken. This is useful when restoring to a cluster having more data nodes than the original. See Restoring to More Nodes Than the Original, for more information.

Tables using one of the two per-node-group settings for the fragment count type can also be fully replicated. This requires that the table's fragment count type is <code>ONE_PER_NODE_GROUP</code> or <code>ONE_PER_LDM_PER_NODE_GROUP</code>, and can be enabled using the option <code>FULLY_REPLICATED=1</code> within in an <code>NDB_TABLE</code> comment. The option can be enabled by default for all new <code>NDB</code> tables using the <code>ndb_fully_replicated</code> system variable added in this release.

Settings for table-level READ_BACKUP are also supported using the COMMENT="NDB_TABLE=..." syntax. It is also possible (and often preferable) to set multiple options in one comment within a

single CREATE TABLE or ALTER TABLE statement. For more information and examples, see Setting NDB Comment Options.

This release also introduces the ndb_data_node_neighbour system variable, which is intended to be employed with transaction hinting (and fully-replicated tables), and to provide the current SQL node with the ID of a "nearby" data node to use. See the description of this variable in the documentation for more information. (WL #9018, WL #9019, WL #9020)

References: See also: Bug #18435416, Bug #11762155, Bug #54717.

Bugs Fixed

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

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

- Important Change: When started with the --initialize option, mysqld no longer enables the NDBCLUSTER storage engine plugin. This change was needed to prevent attempted initialization of system databases as distributed (rather than as specific to individual SQL nodes), which could result in a metadata lock deadlock. This fix also brings the behavior of --initialize in this regard into line with that of the discontinued --bootstrap option, which started a minimal mysqld instance without enabling NDB. (Bug #22758238)
- **Performance:** A performance problem was found in an internal polling method do_poll() where the polling client did not check whether it had itself been woken up before completing the poll. Subsequent analysis showed that it is sufficient that only some clients in the polling queue receive data. do_poll() can then signal these clients and give up its polling rights, even if the maximum specified wait time (10 ms) has not expired.

This change allows do_poll() to continue polling until either the maximum specified wait time has expired, or the polling client itself has been woken up (by receiving what it was waiting for). This avoids unnecessary thread switches between client threads and thus reduces the associated overhead by as much as 10% in the API client, resulting in a significant performance improvement when client threads perform their own polling. (Bug #81229, Bug #23202735)

- macOS: On OS X, ndb_config failed when an empty string was used for the --host option. (Bug #80689, Bug #22908696)
- Microsoft Windows: ndb_mgmd failed to start on 32-bit Windows platforms, due to an issue with calling dynamically loaded functions; such issues were also likely to occur with other NDB programs using ndb_init(). It was found that all of the functions used are already supported in targeted versions of Windows, so this problem is fixed by removing the dynamic loading of these functions and using the versions provided by the Windows header files instead. (Bug #80876, Bug #23014820)
- Microsoft Windows: When building MySQL NDB Cluster on Windows using more than one parallel build job it was sometimes possible for the build to fail because host_info.exe could not be installed. To fix this problem, the install_mcc target is now always built prior to the host_info target. (Bug #80051, Bug #22566368)
- **Microsoft Windows:** Performing ANALYZE TABLE on a table having one or more indexes caused ndbmtd to fail with an *InvalidAttrInfo* error due to signal corruption. This issue occurred consistently on Windows, but could also be encountered on other platforms. (Bug #77716, Bug #21441297)

- **Solaris:** The ndb_print_file utility failed consistently on Solaris 9 for SPARC. (Bug #80096, Bug #22579581)
- NDB Disk Data: The following improvements were made to logging during restarts by data nodes using MySQL NDB Cluster Disk Data:
 - The total amount of undo log to be applied by the data node is now provided as the total number of pages present in the log. This is a worst case estimate.
 - Progress information is now provided at regular intervals (once for each 30000 records) as the
 undo log is applied. This information is supplied as the number of records and number of undo log
 pages applied so far during the current restart.

(Bug #22513381)

- NDB Cluster APIs: Deletion of Ndb objects used a dispoportionately high amount of CPU. (Bug #22986823)
- NDB Cluster APIs: Executing a transaction with an NdbIndexOperation based on an obsolete unique index caused the data node process to fail. Now the index is checked in such cases, and if it cannot be used the transaction fails with an appropriate error. (Bug #79494, Bug #22299443)
- Reserved send buffer for the loopback transporter, introduced in MySQL NDB Cluster 7.4.8 and used by API and management nodes for administrative signals, was calculated incorrectly. (Bug #23093656, Bug #22016081)

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

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

References: See also: Bug #23221573.

- Following the forced shutdown of one of the 2 data nodes in a cluster where NoOfReplicas=2, the
 other data node shut down as well, due to arbitration failure. (Bug #23006431)
- Aborting a CREATE LOGFILE GROUP statement which had failed due to lack of shared global memory was not performed correctly, causing node failure. In addition, the transaction in which this occurred was not rolled back correctly, also causing any subsequent CREATE LOGFILE GROUP to fail. (Bug #22982618)
- The ndbinfo.tc_time_track_stats table uses histogram buckets to give a sense of the distribution of latencies. The sizes of these buckets were also reported as HISTOGRAM BOUNDARY INFO messages during data node startup; this printout was redundant and so has been removed. (Bug #22819868)
- Online upgrades from previous versions of MySQL NDB Cluster to MySQL NDB Cluster 7.5 were not
 possible due to missing entries in the matrix used to test upgrade compatibility between versions.
 (Bug #22024947)
- A failure occurred in DBTUP in debug builds when variable-sized pages for a fragment totalled more than 4 GB. (Bug #21313546)
- Restoration of metadata with ndb_restore -m occasionally failed with the error message Failed to create index... when creating a unique index. While disgnosing this problem, it was found that the internal error PREPARE_SEIZE_ERROR (a temporary error) was reported as an unknown error. Now in such cases, ndb_restore retries the creation of the unique index, and PREPARE_SEIZE_ERROR is reported as NDB Error 748 Busy during read of event table. (Bug #21178339)

References: See also: Bug #22989944.

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

References: See also: Bug #23343739.

- The following improvements were made to the data node error logging mechanism:
 - Increased the message slot size 499 bytes to 999 bytes to prevent log messages from overwriting one another or from being truncated.
 - Added a Trace file name field to the output. This field contains the trace file name (without any path) and trace file number for the thread causing the trace.
 - ndbmtd trace files are also now shown in the error log.

(Bug #21082710)

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

See also The DBDICT Block. (Bug #20368450)

References: See also: Bug #20368354.

- mysql_upgrade failed to upgrade the sys schema if a sys database directory existed but was empty. (Bug #81352, Bug #23249846, Bug #22875519)
- When a write to the ndb_binlog_index table failed during a MySQL Server shutdown, mysqld killed the NDB binary logging thread. (Bug #81166, Bug #23142945)
- Memory associated with table descriptions was not freed by the internal table information method NdbDictInterface::parseTableInfo(). (Bug #81141, Bug #23130084)
- Improved memory usage by the internal TransporterFacade constructor when performing mutex array initialization. (Bug #81134, Bug #23127824)
- Fixed a memory leak that occurred when an error was raised in ha_ndbcluster::get_metadata() or one of the functions which this method calls. (Bug #81045, Bug #23089566)
- An internal function used to validate connections failed to update the connection count when creating a new Ndb object. This had the potential to create a new Ndb object for every operation validating the connection, which could have an impact on performance, particularly when performing schema operations. (Bug #80750, Bug #22932982)
- A table scan on an NDB table using neither an ordered index nor any Disk Data columns normally
 uses an ACC scan. If this happened while scanning an unique but unordered index which shrank (due
 to rows being deleted) after the scan started and then grew again (rows inserted), a single row that
 had been neither deleted nor inserted could be scanned twice. (Bug #80733, Bug #22926938)
- Starting a backup in the ndb_mgm client after creating a large number of tables caused a forced shutdown of the cluster. (Bug #80640, Bug #228849958)

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

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

- The MySQL NDB Cluster Auto-Installer failed to work in various ways on different platforms. (Bug #79853, Bug #22502247)
- The internal function ndbrequire(), which, like assert(), evaluates a given expression and terminates the process if the expression does not evaluate as true, now includes the failed expression in its output to the error logs. (Bug #77021, Bug #21128318)
- Trying to drop a table during an ongoing backup failed with the error message Unknown table; now, it fails with Unable to alter table as backup is in progress. (Bug #47301, Bug #11755512)

References: See also: Bug #44695, Bug #11753280.

Changes in MySQL NDB Cluster 7.5.1 (5.7.11-ndb-7.5.1) (2016-03-18, Development Milestone)

MySQL NDB Cluster 7.5.1 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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 5.7.11 (see Changes in MySQL 5.7.11 (2016-02-05, General Availability)).

- · Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

• Important Change: When creating or adding columns to NDB tables, the default value used for both the COLUMN_FORMAT and ROW_FORMAT options is now DYNAMIC rather than FIXED.

This change does not affect the row format or column format used by existing tables. New columns added to such tables use the new defaults, and existing columns are changed to use these as well, provided that the ALTER TABLE statement in question uses ALGORITHM=COPY. Note that this cannot be done implicitly if mysqld is run with --ndb-allow-copying-alter-table=FALSE (the default is TRUE).

A new MySQL server option --ndb-default-column-format is added for backwards compatibility; set this to FIXED to force the old defaults to be used for COLUMN_FORMAT and ROW_FORMAT.



Note

This behavior change is superseded in MySQL NDB Cluster 7.5.4.

(WL #8573)

• Scans have been improved by replacing the DIH_SCAN_GET_NODES_REQ signal, formerly used for communication between the DBTC and DBDIH kernel blocks in NDB, with the DIGETNODESREQ signal, which supports direct execution and allows for the elimination of DIH_SCAN_GET_NODES_REF and DIH_SCAN_GET_NODES_CONF, as well as for DIH_SCAN_TAB_REQ and DIH_SCAN_TAB_COMPLETE_REP signals to EXECUTE_DIRECT. This enables enable higher scalability of data nodes when used for scan operations by decreasing the use of CPU resources for scan operations by an estimated five percent in some cases. This should also improve response times, which could help prevent issues with overload of the main threads.

As part of these changes, scans made in the BACKUP kernel block have also been improved and made more efficient. (WL #8937)

References: See also: Bug #80640, Bug #22884995.

- The following improvements have been made to event buffer reporting in the cluster log:
 - · Each report now identifies the API node that sent it.
 - The fields shown in the report have been improved and expanded. Percentages are now better specified and used only when appropriate. For improved clarity, the apply_epoch and latest_epoch fields have been renamed to latest_consumed_epoch and latest_buffered_epoch, respectively. The ID of the Ndb object serving as the source of the report is now shown, as is the reason for making the report (as the report reason field).
 - The frequency of unnecessary reporting has been reduced by limiting reports to those showing only significant changes in event buffer usage.
 - The MGM API adds a new NDB_LE_EventBufferStatus2 event type to handle the additional information provided by the new event buffer reporting. The NDB_LE_EventBufferStatus event type used in older versions of MySQL NDB Cluster is now deprecated, and will eventually be removed.

For more information, see Event Buffer Reporting in the Cluster Log, as well as The ndb_logevent Structure. (WL #8626)

Bugs Fixed

- Important Change: The minimum value for the BackupDataBufferSize data node configuration parameter has been lowered from 2 MB to 512 KB. The default and maximum values for this parameter remain unchanged. (Bug #22749509)
- OS X: Processing of local checkpoints was not handled correctly on Mac OS X, due to an uninitialized variable. (Bug #80236, Bug #22647462)
- Microsoft Windows: Compilation of MySQL with Visual Studio 2015 failed in ConfigInfo.cpp, due to a change in Visual Studio's handling of spaces and concatenation. (Bug #22558836, Bug #80024)
- Microsoft Windows: When setting up event logging for ndb_mgmd on Windows, MySQL NDB Cluster tries to add a registry key to HKEY_LOCAL_MACHINE, which fails if the user does not have access to the registry. In such cases ndb_mgmd logged the error Could neither create or open key, which is not accurate and which can cause confusion for users who may not realize that file logging is available and being used. Now in such cases, ndb_mgmd logs a warning Could not create or access the registry key needed for the application to log to the Windows EventLog. Run the application with sufficient privileges once to create the key, or add the key manually, or turn off logging for that application. An error (as opposed to a warning) is now reported in such cases only if there is no available output at all for ndb_mgmd event logging. (Bug #20960839)

- **Microsoft Windows:** MySQL NDB Cluster did not compile correctly with Microsoft Visual Studio 2015, due to a change from previous versions in the VS implementation of the _vsnprintf() function. (Bug #80276, Bug #22670525)
- During node failure handling, the request structure used to drive the cleanup operation was not
 maintained correctly when the request was executed. This led to inconsistencies that were harmless
 during normal operation, but these could lead to assertion failures during node failure handling, with
 subsequent failure of additional nodes. (Bug #22643129)
- The previous fix for a lack of mutex protection for the internal TransporterFacade::deliver_signal() function was found to be incomplete in some cases. (Bug #22615274)

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

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

References: See also: Bug #22389653.

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

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

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

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

- Optimization of signal sending by buffering and sending them periodically, or when the buffer became full, could cause SUB_GCP_COMPLETE_ACK signals to be excessively delayed. Such signals are sent for each node and epoch, with a minimum interval of TimeBetweenEpochs; if they are not received in time, the SUMA buffers can overflow as a result. The overflow caused API nodes to be disconnected, leading to current transactions being aborted due to node failure. This condition made it difficult for long transactions (such as altering a very large table), to be completed. Now in such cases, the ACK signal is sent without being delayed. (Bug #18753341)
- When setting CPU spin time, the value was needlessly cast to a boolean internally, so that setting it to any nonzero value yielded an effective value of 1. This issue, as well as the fix for it, apply both to

setting the SchedulerSpinTimer parameter and to setting spintime as part of a ThreadConfig parameter value. (Bug #80237, Bug #22647476)

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

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

- Suppressed a CMake warning that was caused by use of an incorrectly quoted variable name. (Bug #80066, Bug #22572632)
- When using CREATE INDEX to add an index on either of two NDB tables sharing circular foreign keys, the query succeeded but a temporary table was left on disk, breaking the foreign key constraints. This issue was also observed when attempting to create an index on a table in the middle of a chain of foreign keys—that is, a table having both parent and child keys, but on different tables. The problem did not occur when using ALTER TABLE to perform the same index creation operation; and subsequent analysis revealed unintended differences in the way such operations were performed by CREATE INDEX.

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

• The PortNumber SCI, SHM, and TCP configuration parameters, which were deprecated in MySQL 5.1.3, have now been removed and are no longer accepted in configuration files.

This change does not affect the PortNumber management node configuration parameter, whose behavior remains unaltered. (Bug #77405, Bug #21280456)

Changes in MySQL NDB Cluster 7.5.0 (5.7.10-ndb-7.5.0) (2016-02-05, Development Milestone)

MySQL NDB Cluster 7.5.0 is a new release of MySQL NDB Cluster 7.5, based on MySQL Server 5.7 and including features in version 7.5 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 7.5. MySQL NDB Cluster 7.5 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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 5.7.10 (see Changes in MySQL 5.7.10 (2015-12-07, General Availability)).

- · Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Important Change: Previously, the ndbinfo information database included lookup tables that used
 the MyISAM storage engine. This dependency on MyISAM has now been removed. (Bug #20075747,
 WL #7575)
- Important Change: Previously, the NDB scheduler always optimized for speed against throughput in a predetermined manner (this was hard coded); this balance can now be set using the SchedulerResponsiveness data node configuration parameter. This parameter accepts an integer in the range of 0-10 inclusive, with 5 as the default. Higher values provide better response

times relative to throughput. Lower values provide increased throughput, but impose longer response times. (Bug #78531, Bug #21889312)

• Important Change: A number of MySQL NDB Cluster data node configuration parameters were deprecated in earlier versions of MySQL NDB Cluster, and have been removed with this release. These parameters include Id, NoOfDiskPagesToDiskDuringRestartTUP, NoOfDiskPagesToDiskDuringRestartACC, NoOfDiskPagesToDiskAfterRestartACC, NoOfDiskPagesToDiskAfterRestartTUP, ReservedSendBufferMemory, MaxNoOfIndexes, and Discless (use Diskless instead), as well as DiskCheckpointSpeed and DiskCheckpointSpeedInRestart. The archaic and unused ByteOrder computer configuration parameter has also been removed, as well as the unused MaxNoOfSavedEvents management node confugration parameter. These parameters are no longer supported; most of them already did not have (or no longer had) any effect. Trying to use any of these parameters in a MySQL NDB Cluster configuration file now results in an error.

For more information, see What is New in NDB Cluster 7.5. (Bug #77404, Bug #21280428)

- Important Change: The ndbinfo database can now provide default and current information about MySQL NDB Cluster node configuration parameters as a result of the following changes:
 - 1. The config_params table has been enhanced with additional columns providing information about each configuration parameter, including its type, default, and maximum and minimum values (where applicable).
 - 2. A new config_values table has been added. A row in this table shows the current value of a parameter on a given node.

You can obtain values of MySQL NDB Cluster configuration parameters by name using a join on these two tables such as the one shown here:

(Bug #71587, Bug #18183958, WL #8703)

• Important Change: The ExecuteOnComputer configuration parameter for management, data, and API nodes is now deprecated, and is subject to removal in a future MySQL NDB Cluster version. For all types of MySQL NDB Cluster nodes, you should now use the HostName parameter exclusively for identifying hosts in the cluster configuration file.

This information is also now displayed in the output of ndb_config --configinfo --xml. (Bug #53052, Bug #11760628)

- NDB Replication: Normally, RESET SLAVE causes all entries to be deleted from the mysql.ndb_apply_status table. This release adds the ndb_clear_apply_status system variable, which makes it possible to override this behavior. This variable is ON by default; setting it to OFF keeps RESET SLAVE from purging the ndb_apply_status table. (Bug #12630403)
- Deprecated MySQL NDB Cluster node configuration parameters are now indicated as such by ndb_config --configinfo --xml. For each parameter currently deprecated, the corresponding <param/> tag in the XML output now includes the attribute deprecated="true". (Bug #21127135)
- Added the --ndb-cluster-connection-pool-nodeids option for mysqld, which can be used
 to specify a list of nodes by node ID for connection pooling. The number of node IDs in the list must
 equal the value set for --ndb-cluster-connection-pool. (Bug #19521789)

- Added the PROMPT command in the ndb_mgm client. This command has the syntax PROMPT string, which sets the client's prompt to string. Issuing the command without an argument causes the prompt to be reset to the default (ndb_mgm>). See Commands in the NDB Cluster Management Client, for more information. (Bug #18421338)
- When the --database option has not been specified for ndb_show_tables, and no tables are
 found in the TEST_DB database, an appropriate warning message is now issued. (Bug #50633, Bug
 #11758430)
- The NDB storage engine now uses the improved records-per-key interface for index statistics introduced for the optimizer in MySQL 5.7. Some improvements due to this change are listed here:
 - The optimizer can now choose better execution plans for queries on NDB tables in many cases where a less optimal join index or table join order would previously have been chosen.
 - EXPLAIN now provides more accurate row estimates than previously.
 - Improved cardinality estimates can be obtained from SHOW INDEX.

(WL #8165)

Bugs Fixed

- Incompatible Change; NDB Cluster APIs: The pollEvents2() method now returns -1, indicating an error, whenever a negative value is used for the time argument. (Bug #20762291)
- Important Change; NDB Cluster APIs: Ndb::pollEvents() is now compatible with the TE_EMPTY, TE_INCONSISTENT, and TE_OUT_OF_MEMORY event types introduced in MySQL NDB Cluster 7.4.3. For detailed information about this change, see the description of this method in the MySQL NDB Cluster API Developer Guide. (Bug #20646496)
- Important Change; NDB Cluster APIs: Added the method

 Ndb::isExpectingHigherQueuedEpochs() to the NDB API to detect when additional, newer
 event epochs were detected by pollEvents2().

The behavior of Ndb::pollEvents() has also been modified such that it now returns NDB_FAILURE_GCI (equal to ~(Uint64) 0) when a cluster failure has been detected. (Bug #18753887)

• Important Change; NDB Cluster APIs: To release the memory used for dropped event operations, the event API formerly depended on pollEvents() and nextEvent() to consume all events possibly referring to the dropped events. This dependency between dropEventOperation() and the first two methods required the entire event buffer to be read before attempting to release event operation memory (that is, until successive calls to pollEvents() and nextEvent() returned no more events).

A related cleanup issue arose following the reset of the event buffer (when all event operations had previously been dropped), and the event buffer was truncated by the first createEventOperation() call subsequent to the reset.

To fix these problems, the event buffer is now cleared when the last event operation is dropped, rather than waiting for a subsequent create operation which might or might not occur. Memory taken up by dropped event operations is also now released when the event queue has been cleared, which removes the hidden requirement for consuming all events to free up memory. In addition, event operation memory is now released as soon as all events referring to the operation have been consumed, rather than waiting for the entire event buffer to be consumed. (Bug #78145, Bug #21661297)

• Important Change; NDB Cluster APIs: The MGM API error-handling functions ndb_mgm_get_latest_error(), ndb_mgm_get_latest_error_msg(), and ndb_mgm_get_latest_error_desc() each failed when used with a NULL handle. You should

note that, although these functions are now null-safe, values returned in this case are arbitrary and not meaningful. (Bug #78130, Bug #21651706)

- Important Change; NDB Cluster APIs: The following NDB API methods were not actually implemented and have been removed from the sources:
 - Datafile methods: getNode(), setNode(), and getFileNo()
 - Undofile methods: getNode(), setNode(), and getFileNo()
 - Table methods: getObjectType() and setObjectType()
- **Important Change:** The options controlling behavior of NDB programs with regard to the number and timing of successive attempts to connect to a management server have changed as listed here:
 - The minimum value for the --connect-retry-delay option common to all NDB programs has been changed from 0 to 1; this means that all NDB programs now wait at least 1 second between successive connection attempts, and it is no longer possible to set a waiting time equal to 0.
 - The semantics for the --connect-retries option have changed slightly, such that the value of this option now sets the number of times an NDB program tries to connect to a management server. Setting this option to 0 now causes the program to attempt the connection indefinitely, until it either succeeds or is terminated by other means (such as kill).
 - In addition, the default for the --connect-retries option for the ndb_mgm client has been changed from 3 to 12, so that the minimum, maximum, and default values for this option when used with ndb_mgm are now exactly the same as for all other NDB programs.

The ndb_mgm --try-reconnect option, although deprecated in MySQL NDB Cluster 7.4, continues to be supported as a synonym for ndb_mgm --connect-retries to provide backwards compatibility. The default value for --try-reconnect has also been changed from 3 to 12, respectively, so that this option continues to behave in the exactly in the same way as --connect-retries.

(Bug #22116937)

- Important Change: In previous versions of MySQL NDB Cluster, other DDL operations could not be part of ALTER ONLINE TABLE ... RENAME (This was disallowed by the fix for BUG#16021021.) MySQL NDB Cluster 7.5 makes the following changes:
 - Support for the ONLINE and OFFLINE keywords, which was deprecated in MySQL NDB Cluster 7.3, is now removed, and use of these now causes a syntax error; the NDB storage engine now accepts *only* ALGORITHM = DEFAULT, ALGORITHM = COPY, and ALGORITHM = INPLACE to specify whether the ALTER operation is copying or in-place, just as in the standard MySQL Server.
 - NDB now allows ALTER TABLE ... ALGORITHM=COPYING RENAME.

(Bug #20804269, Bug #76543, Bug #20479917, Bug #75797)

References: See also: Bug #16021021.

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

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

When the binary log inject thread receives a group of one or more failure events, it drops all other existing event operations and expects no more events from the utility thread until it has handled all of the failure events and then restarted itself. However, it was possible for the utility thread to continue attempting binary log setup while the injector thread was handling failures and thus attempting to create the schema distribution tables as well as event subscriptions on these tables. If the creation of these tables and event subscriptions occurred during this time, the binary log injector thread's expectation that there were no further event operations was never met; thus, the injector thread never restarted, and NDB tables remained in read-only as described previously.

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

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

References: See also: Bug #20646496.

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

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

- NDB Cluster APIs: Creation and destruction of Ndb_cluster_connection objects by multiple
 threads could make use of the same application lock, which in some cases led to failures in the
 global dictionary cache. To alleviate this problem, the creation and destruction of several internal
 NDB API objects have been serialized. (Bug #20636124)
- NDB Cluster APIs: When an Ndb object created prior to a failure of the cluster was reused, the
 event queue of this object could still contain data node events originating from before the failure.
 These events could reference "old" epochs (from before the failure occurred), which in turn could
 violate the assumption made by the nextEvent() method that epoch numbers always increase.
 This issue is addressed by explicitly clearing the event queue in such cases. (Bug #18411034)

References: See also: Bug #20888668.

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

Introduction of transporter polling also revealed a problem with missing mutex protection in the ndbcluster_binlog handler, which has been added as part of this fix. (Bug #79311, Bug #20957068, Bug #22224571, WL #8627)

NDB Cluster APIs: After the initial restart of a node following a cluster failure, the cluster failure
event added as part of the restart process was deleted when an event that existed prior to the
restart was later deleted. This meant that, in such cases, an Event API client had no way of knowing
that failure handling was needed. In addition, the GCI used for the final cleanup of deleted event

operations, performed by pollEvents() and nextEvent() when these methods have consumed all available events, was lost. (Bug #78143, Bug #21660947)

- A serious regression was inadvertently introduced in MySQL NDB Cluster 7.4.8 whereby local checkpoints and thus restarts often took much longer than expected. This occurred due to the fact that the setting for MaxDiskWriteSpeedOwnRestart was ignored during restarts and the value of MaxDiskWriteSpeedOtherNodeRestart, which is much lower by default than the default for MaxDiskWriteSpeedOwnRestart, was used instead. This issue affected restart times and performance only and did not have any impact on normal operations. (Bug #22582233)
- The epoch for the latest restorable checkpoint provided in the cluster log as part of its reporting for EventBufferStatus events (see NDB Cluster: Messages in the Cluster Log) was not well defined and thus unreliable; depending on various factors, the reported epoch could be the one currently being consumed, the one most recently consumed, or the next one queued for consumption.

This fix ensures that the latest restorable global checkpoint is always regarded as the one that was most recently completely consumed by the user, and thus that it was the latest restorable global checkpoint that existed at the time the report was generated. (Bug #22378288)

Added the --ndb-allow-copying-alter-table option for mysqld. Setting this option (or the
equivalent system variable ndb_allow_copying_alter_table) to OFF keeps ALTER TABLE
statements from performing copying operations. The default value is ON. (Bug #22187649)

References: See also: Bug #17400320.

- Attempting to create an NDB table having greater than the maximum supported combined width for all BIT columns (4096) caused data node failure when these columns were defined with COLUMN_FORMAT DYNAMIC. (Bug #21889267)
- Creating a table with the maxmimum supported number of columns (512) all using COLUMN_FORMAT DYNAMIC led to data node failures. (Bug #21863798)
- In a MySQL NDB Cluster with multiple LDM instances, all instances wrote to the node log, even inactive instances on other nodes. During restarts, this caused the log to be filled with messages from other nodes, such as the messages shown here:

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

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

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

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

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

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

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

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

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

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

References: See also: Bug #20481140.

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

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

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

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

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

 Memory allocated when obtaining a list of tables or databases was not freed afterward. (Bug #20234681, Bug #74510)

References: See also: Bug #18592390, Bug #72322.

• Added the BackupDiskWriteSpeedPct data node parameter. Setting this parameter causes the data node to reserve a percentage of its maximum write speed (as determined by the value of MaxDiskWriteSpeed) for use in local checkpoints while performing a backup.

BackupDiskWriteSpeedPct is interpreted as a percentage which can be set between 0 and 90 inclusive, with a default value of 50. (Bug #20204854)

References: See also: Bug #21372136.

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

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

Prior to upgrading, this issue can be worked around by executing SELECT TABLE_NAME, TABLE_SCHEMA FROM INFORMATION_SCHEMA.TABLES WHERE ENGINE = 'NDBCLUSTER'

on all SQL nodes following the restore operation, before executing any other statements. (Bug #18075170)

- Using ndb_mgm STOP -f to force a node shutdown even when it triggered a complete shutdown of the cluster, it was possible to lose data when a sufficient number of nodes were shut down, triggering a cluster shutodwn, and the timing was such that SUMA handovers had been made to nodes already in the process of shutting down. (Bug #17772138)
- When using a sufficiently large value for TransactionDeadlockDetectionTimeout
 and the default value for sort_buffer_size, executing SELECT * FROM
 ndbinfo.cluster_operations ORDER BY transid with multiple concurrent conflicting or
 deadlocked transactions, each transaction having several pending operations, caused the SQL node
 where the query was run to fail. (Bug #16731538, Bug #67596)
- The ndbinfo.config_params table is now read-only. (Bug #11762750, Bug #55383)
- NDB failed during a node restart due to the status of the current local checkpoint being set but not as
 active, even though it could have other states under such conditions. (Bug #78780, Bug #21973758)
- ndbmtd checked for signals being sent only after a full cycle in run_job_buffers, which is performed for all job buffer inputs. Now this is done as part of run_job_buffers itself, which avoids executing for extended periods of time without sending to other nodes or flushing signals to other threads. (Bug #78530, Bug #21889088)
- When attempting to enable index statistics, creation of the required system tables, events and event subscriptions often fails when multiple mysqld processes using index statistics are started concurrently in conjunction with starting, restarting, or stopping the cluster, or with node failure handling. This is normally recoverable, since the affected mysqld process or processes can (and do) retry these operations shortly thereafter. For this reason, such failures are no longer logged as warnings, but merely as informational events. (Bug #77760, Bug #21462846)
- It was possible to end up with a lock on the send buffer mutex when send buffers became a limiting resource, due either to insufficient send buffer resource configuration, problems with slow or failing communications such that all send buffers became exhausted, or slow receivers failing to consume what was sent. In this situation worker threads failed to allocate send buffer memory for signals, and attempted to force a send in order to free up space, while at the same time the send thread was busy trying to send to the same node or nodes. All of these threads competed for taking the send buffer mutex, which resulted in the lock already described, reported by the watchdog as Stuck in Send. This fix is made in two parts, listed here:
 - The send thread no longer holds the global send thread mutex while getting the send buffer mutex; it now releases the global mutex prior to locking the send buffer mutex. This keeps worker threads from getting stuck in send in such cases.
 - 2. Locking of the send buffer mutex done by the send threads now uses a try-lock. If the try-lock fails, the node to make the send to is reinserted at the end of the list of send nodes in order to be retried later. This removes the Stuck in Send condition for the send threads.

(Bug #77081, Bug #21109605)

Release Series Changelogs: MySQL NDB Cluster 7.5

This section contains unified changelog information for the MySQL NDB Cluster 7.5 release series.

For changelogs covering individual MySQL NDB Cluster 7.5 releases, see NDB Cluster Release Notes.

For general information about features added in MySQL NDB Cluster 7.5, see What is New in NDB Cluster 7.5.

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 7.5.34 (5.7.44-ndb-7.5.34) (2024-04-30, General Availability)

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)

Changes in MySQL NDB Cluster 7.5.33 (5.7.44-ndb-7.5.33) (2024-01-16, General Availability)

Compilation Notes

 Microsoft Windows: NDB Cluster did not compile correctly using Visual Studio 2022. (Bug #35967676)

Changes in MySQL NDB Cluster 7.5.31 (5.7.43-ndb-7.5.31) (2023-07-18, General Availability)

- · Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

NDB 7.5 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 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)

Changes in MySQL NDB Cluster 7.5.30 (5.7.42-ndb-7.5.30) (2023-04-19, General Availability)

- 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)
- 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)

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 #90360, Bug
#27834961)

References: See also: Bug #85411, Bug #25703113.

Changes in MySQL NDB Cluster 7.5.29 (5.7.41-ndb-7.5.29) (2023-01-18, General Availability)

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 <code>CopyFrag</code> 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 <code>CopyFrag</code> 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)
- 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 7.5.28 (5.7.40-ndb-7.5.28) (2202-10-12, General Availability)

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 7.5.27 (5.7.39-ndb-7.5.27) (2022-07-27, General Availability)

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 7.5.26 (5.7.38-ndb-7.5.26) (2022-04-27, General Availability)

- · 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 <code>Dbacc::getElement()</code>, and the root cause identified to be the timing of resizing the structures used for storing elements by <code>DBACC</code>, 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 <code>computeXorChecksum()</code> 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 <code>xor</code> 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 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.

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 7.5.25 (5.7.37-ndb-7.5.25) (2022-01-19, General Availability)

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)
- 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)
- 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.

- 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 7.5.24 (5.7.36-ndb-7.5.24) (2021-10-20, General Availability)

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 7.5.23 (5.7.35-ndb-7.5.23) (2021-07-21, General Availability)

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

Changes in MySQL NDB Cluster 7.5.22 (5.7.34-ndb-7.5.22) (2021-04-21, General Availability)

Bugs Fixed

- 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)
- 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)

Changes in MySQL NDB Cluster 7.5.21 (5.7.33-ndb-7.5.21) (2021-01-19, General Availability)

- Deprecation and Removal Notes
- Bugs Fixed

Deprecation and Removal Notes

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

References: See also: Bug #31888835.

• **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)
- 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 7.5.20 (5.7.32-ndb-7.5.20) (2020-10-20, General Availability)

• Deprecation and Removal Notes

· Bugs Fixed

Deprecation and Removal Notes

- NDB Cluster APIs: Support for Node.js has been removed in this release.
 - Node.js continues to be supported in NDB Cluster 8.0 only. (Bug #31781948)
- NDB Client Programs: Effective with this release, the MySQL NDB Cluster Auto-Installer (ndb_setup.py) has been deprecated and is subject to removal in a future version of NDB Cluster. (Bug #31888835)
- **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)

Changes in MySQL NDB Cluster 7.5.19 (5.7.31-ndb-7.5.19) (2020-07-14, General Availability)

Bugs Fixed

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

requests. While the copy operation is in progress, user-level SUB_START and SUB_STOP requests are blocked using a DICT lock.

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

- 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)

• 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)

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 7.5.18 (5.7.30-ndb-7.5.18) (2020-04-28, General Availability)

- · 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)

• 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 7.5.17 (5.7.29-ndb-7.5.17) (2020-01-14, General Availability)

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)

- 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)
- 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)
- 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 <code>ODirect</code> was enabled for an EOF backup data file write whose size was less than 512 bytes and the backup was in the <code>STOPPING</code> 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 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 7.5.16 (5.7.28-ndb-7.5.16) (2019-10-15, General Availability)

- · 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)

Bugs Fixed

- 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 <code>java.lang.IllegalArgumentException: non-public interface is not defined</code> 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 <code>DomainTypeHandlerImpl::Finalizable</code> 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 <code>Finalization</code> 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)
- 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.

- 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)
- 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)

Changes in MySQL NDB Cluster 7.5.15 (5.7.27-ndb-7.5.15) (2019-07-23, General Availability)

- · Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

• Building with CMake3 is now supported by the compile-cluster script included in the NDB source distribution. (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)
- 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)
- 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 resourceconstrained environments. (Bug #29528188)
- 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 7.5.14 (5.7.26-ndb-7.5.14) (2019-04-26, General Availability)

Bugs Fixed

- NDB Disk Data: NDB did not validate MaxNoOfOpenFiles in relation to InitialNoOfOpenFiles correctly, leading data nodes to fail with an error message that did not make the nature of the problem clear 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.

- 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) 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.

 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".

(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 7.5.13 (5.7.25-ndb-7.5.13) (2019-01-22, General Availability)

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.

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)

- Executing SELECT * FROM INFORMATION_SCHEMA.TABLES caused SQL nodes to restart in some cases. (Bug #27613173)
- When tables with BLOB columns were dropped and then re-created with a different number of BLOB columns the event definitions for monitoring table changes could become inconsistent in certain error situations involving communication errors when the expected cleanup of the corresponding events was not performed. In particular, when the new versions of the tables had more BLOB columns than the original tables, some events could be missing. (Bug #27072756)
- When running a cluster with 4 or more data nodes under very high loads, data nodes could sometimes fail with Error 899 Rowid already allocated. (Bug #25960230)
- When starting, a data node copies metadata, while a local checkpoint updates metadata. To avoid any conflict, any ongoing LCP activity is paused while metadata is being copied. An issue arose when a local checkpoint was paused on a given node, and another node that was also restarting checked for a complete LCP on this node; the check actually caused the LCP to be completed before copying of metadata was complete and so ended the pause prematurely. Now in such cases, the LCP completion check waits to complete a paused LCP until copying of metadata is finished and the pause ends as expected, within the LCP in which it began. (Bug #24827685)
- Asynchronous disconnection of mysqld from the cluster caused any subsequent attempt to start an NDB API transaction to fail. If this occurred during a bulk delete operation, the SQL layer called HA::end_bulk_delete(), whose implementation by ha_ndbcluster assumed that a transaction had been started, and could fail if this was not the case. This problem is fixed by checking that the transaction pointer used by this method is set before referencing it. (Bug #20116393)
- 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 7.5.12 (5.7.24-ndb-7.5.12) (2018-10-23, General Availability)

Bugs Fixed

- **Packaging:** Expected NDB header files were in the devel RPM package instead of libndbclient-devel. (Bug #84580, Bug #26448330)
- MySQL NDB ClusterJ: When a table containing a BLOB or a TEXT field was being queried with ClusterJ for a record that did not exist, an exception ("The method is not valid in current blob state") was thrown. (Bug #28536926)
- MySQL NDB ClusterJ: A NullPointerException was thrown when a full table scan was performed with ClusterJ on tables containing either a BLOB or a TEXT field. It was because the proper object initializations were omitted, and they have now been added by this fix. (Bug #28199372, Bug #91242)
- When the SUMA kernel block receives a SUB_STOP_REQ signal, it executes the signal then replies with SUB_STOP_CONF. (After this response is relayed back to the API, the API is open to send more SUB_STOP_REQ signals.) After sending the SUB_STOP_CONF, SUMA drops the subscription if no subscribers are present, which involves sending multiple DROP_TRIG_IMPL_REQ messages to DBTUP. LocalProxy can handle up to 21 of these requests in parallel; any more than this are queued

in the Short Time Queue. When execution of a DROP_TRIG_IMPL_REQ was delayed, there was a chance for the queue to become overloaded, leading to a data node shutdown with Error in short time queue.

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

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

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

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

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

- 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)
- 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 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.

Changes in MySQL NDB Cluster 7.5.11 (5.7.23-ndb-7.5.11) (2018-07-27, General Availability)

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)
- MySQL NDB ClusterJ: ClusterJ could not be built from source using JDK 9. (Bug #27977985)
- An internal buffer being reused immediately after it had been freed could lead to an unplanned data node shutdown. (Bug #27622643)

References: See also: Bug #28698831.

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

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

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

References: See also: Bug #27566346.

• 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.

- 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.

• 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).

• 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 7.5.10 (5.7.22-ndb-7.5.10) (2018-04-20, General Availability)

Bugs Fixed

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

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

- NDB Cluster APIs: Incorrect results, usually an empty result set, were returned when setBound()
 was used to specify a NULL bound. This issue appears to have been caused by a problem in gcc, limited to cases using the old version of this method (which does not employ NdbRecord), and is fixed by rewriting the problematic internal logic in the old implementation. (Bug #89468, Bug #27461752)
- 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)
- 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)
- Under certain conditions, data nodes restarted unnecessarily during execution of ALTER TABLE...
 REORGANIZE PARTITION. (Bug #25675481)

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

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

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

References: See also: Bug #20112700.

- ndb_restore sometimes logged data file and log file progress values much greater than 100%.
 (Bug #20989106)
- 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)
- ndb_restore --print-data --hex did not print trailing 0s of LONGVARBINARY values. (Bug #65560, Bug #14198580)

Changes in MySQL NDB Cluster 7.5.9 (5.7.21-ndb-7.5.9) (2018-01-17, General Availability)

Bugs Fixed

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

In cases where closing of the file was slow, it was possible for more than 4 files per redo log part to be open concurrently; since these files were opened using the <code>OM_WRITE_BUFFER</code> 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 <code>OM_WRITE_BUFFER</code> 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)
- In certain circumstances where multiple Ndb objects were being used in parallel from an API node,
 the block number extracted from a block reference in DBLQH was the same as that of a SUMA block
 even though the request was coming from an API node. Due to this ambiguity, DBLQH mistook the
 request from the API node for a request from a SUMA block and failed. This is fixed by checking node
 IDs before checking block numbers. (Bug #88441, Bug #27130570)
- 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)
- 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 <code>glibc</code>. The specific <code>glibc</code> issue being guarded against is encountered only in response to an internal NDB API call (<code>Ndb_UnlockCPU()</code>) 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 <code>glibc</code> version is used. (Bug #87683, Bug #26758939)

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

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 7.5.8 (5.7.20-ndb-7.5.8) (2017-10-18, General Availability)

- · Functionality Added or Changed
- · Bugs Fixed

Functionality Added or Changed

 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)

Bugs Fixed

- 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)

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

Changes in MySQL NDB Cluster 7.5.7 (5.7.19-ndb-7.5.7) (2017-07-19, General Availability)

- · 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)
- 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)

- 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)
- Added the --diff-default option for ndb_config. This option causes the program to print only those parameters having values that differ from their defaults. (Bug #85831, Bug #25844166)
- Added the --query-all option to ndb_config. This option acts much like the --query option
 except that --query-all (short form: -a) dumps configuration information for all attributes at one
 time. (Bug #60095, Bug #11766869)

Bugs Fixed

- 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)

- 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 Cluster APIs: The implementation method NdbDictionary::NdbTableImpl::getColumn(), used from many places in the NDB API where a column is referenced by name, has been made more efficient. This method used a linear search of an array of columns to find the correct column object, which could be inefficient for tables with many columns, and was detected as a significant use of CPU in customer applications. (Ideally, users should perform name-to-column object mapping, and then use column IDs or objects in method calls, but in practice this is not always done.) A less costly hash index implementation, used previously for the name lookup, is reinstated for tables having relatively many columns. (A linear search continues to be used for tables having fewer columns, where the difference in performance is neglible.) (Bug #24829435)
- MySQL NDB ClusterJ: The JTie and NDB JTie tests were skipped when the unit tests for ClusterJ were being run. (Bug #26088583)
- MySQL NDB ClusterJ: Compilation for the tests for NDB JTie failed. It was due to how null
 references were handled, which has been corrected by this fix. (Bug #26080804)
- Backup .log files contained log entries for one or more extra fragments, due to an issue with
 filtering out changes logged by other nodes in the same node group. This resulted in a larger .log
 file and thus use of more resources than necessary; it could also cause problems when restoring,
 since backups from different nodes could interfere with one another while the log was being applied.
 (Bug #25891014)
- When making the final write to a redo log file, it is expected that the next log file is already opened for
 writes, but this was not always the case with a slow disk, leading to node failure. Now in such cases
 NDB waits for the next file to be opened properly before attempting to write to it. (Bug #25806659)

- Data node threads can be bound to a single CPU or a set of CPUs, a set of CPUs being represented internally by NDB as a SparseBitmask. When attempting to lock to a set of CPUs, CPU usage was excessive due to the fact that the routine performing the locks used the mt_thr_config.cpp::do_bind() method, which looks for bits that are set over the entire theoretical range of the SparseBitmask (2³²-2, 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)
- 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.

- Setting NoOfFragmentLogParts such that there were more than 4 redo log parts per local data
 manager led to resource exhaustion and subsequent multiple data node failures. Since this is an
 invalid configuration, a check has been added to detect a configuration with more than 4 redo log
 parts per LDM, and reject it as invalid. (Bug #25333414)
- Execution of an online ALTER TABLE ... REORGANIZE PARTITION statement on an NDB table having a primary key whose length was greater than 80 bytes led to restarting of data nodes, causing the reorganization to fail. (Bug #25152165)
- 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)
- The planned shutdown of an NDB Cluster having more than 10 data nodes was not always performed gracefully. (Bug #20607730)
- Due to a previous issue with unclear separation between the optimize and execute phases when a query involved a GROUP BY, the join-pushable evaluator was not sure whether its optimized query execution plan was in fact pushable. For this reason, such grouped joins were always considered not pushable. It has been determined that the separation issue has been resolved by work already done in MySQL 5.6, and so we now remove this limitation. (Bug #86623, Bug #26239591)
- When deleting all rows from a table immediately followed by DROP TABLE, it was possible that the
 shrinking of the DBACC hash index was not ready prior to the drop. This shrinking is a per-fragment
 operation that does not check the state of the table. When a table is dropped, DBACC releases
 resources, during which the description of the fragment size and page directory is not consistent; this
 could lead to reads of stale pages, and undefined behavior.

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

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

- The internal function <code>execute_signals()</code> in <code>mt.cpp</code> 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, <code>ndbmtd</code> failed with an error. To keep this from occurring, this function now only reads section pointers that are actually passed to it. (Bug #86354, Bug #26092639)
- The ndb_show_tables program --unqualified option did not work correctly when set to 0 (false); this should disable the option and so cause fully qualified table and index names to be printed in the output. (Bug #86017, Bug #25923164)
- When an NDB table with foreign key constraints is created, its indexes are created first, and then, during foreign key creation, these indexes are loaded into the NDB dictionary cache. When a CREATE TABLE statement failed due to an issue relating to foreign keys, the indexes already in the cache were not invalidated. This meant that any subsequent CREATE TABLE with any indexes having the same names as those in the failed statement produced inconsistent results. Now, in such cases, any indexes named in the failed CREATE TABLE are immediately invalidated from the cache. (Bug #85917, Bug #25882950)
- Attempting to execute ALTER TABLE ... ADD FOREIGN KEY when the key to be added had the name of an existing foreign key on the same table failed with the wrong error message. (Bug #85857, Bug #23068914)
- The node internal scheduler (in mt.cpp) collects statistics about its own progress and any outstanding work it is performing. One such statistic is the number of outstanding send bytes, collected in send_buffer::m_node_total_send_buffer_size. This information may later

be used by the send thread scheduler, which uses it as a metric to tune its own send performance versus latency.

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

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

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

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

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

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

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

- Data node binaries for Solaris 11 built using Oracle Developer Studio 12.5 on SPARC platforms failed with bus errors. (Bug #85390, Bug #25695818)
- 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 7.5.6 (5.7.18-ndb-7.5.6) (2017-04-10, General Availability)

- · Functionality Added or Changed
- · Bugs Fixed

Functionality Added or Changed

• Packaging: Yum repo packages are added for EL5, EL6, EL7, and SLES12.

Apt repo packages are added for Debian 7, Debian 8, Ubuntu 14.04, and Ubuntu 16.04

Bugs Fixed

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

This was due to the fact that, when processing an EXPLAIN statement, mysqld calculates the partition ID for a hash value as (hash_value % number_of_partitions), which is correct

only when the table is partitioned by HASH, since other partitioning types use different methods of mapping hash values to partition IDs. This fix replaces the partition ID calculation performed by mysqld with an internal NDB function which calculates the partition ID correctly, based on the table's partitioning type. (Bug #21068548)

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

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

- CPU usage of the data node's main thread by the DBDIH master block as the end of a local
 checkpoint could approach 100% in certain cases where the database had a very large number of
 fragment replicas. This is fixed by reducing the frequency and range of fragment queue checking
 during an LCP. (Bug #25443080)
- The ndb_print_backup_file utility failed when attempting to read from a backup file when the backup included a table having more than 500 columns. (Bug #25302901)

References: See also: Bug #25182956.

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

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

References: See also: Bug #25306089.

- 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)
- 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)

• Ubuntu 14.04 and Ubuntu 16.04 are now supported.

Changes in MySQL NDB Cluster 7.5.5 (5.7.17-ndb-7.5.5) (2017-01-17, General Availability)

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)
- Packaging: The RPM installer for the MySQL NDB Cluster auto-installer package had a dependency on python2-crypt instead of python-crypt. (Bug #24924607)
- Microsoft Windows: Installation failed when the Auto-Installer (ndb_setup.py) was run on a Windows host that used Swedish as the system language. This was due to system messages being issued using the cp1252 character set; when these messages contained characters that did not map directly to 7-bit ASCII (such as the ä character in Tjänsten ... startar), conversion to UTF-8—as expected by the Auto-Installer web client—failed.

This fix has been tested only with Swedish as the system language, but should work for Windows systems set to other European languages that use the cp1252 character set. (Bug #83870, Bug #25111830)

• **ndbinfo Information Database:** A number of inconsistencies and other issues had arisen regarding ndbinfo tables due to manual copying of the table and view definitions in the sources. Now the SQL statements for these are generated, for consistency. (Bug #23305078)

References: See also: Bug #25047951.

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

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

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

- The ndb_show_tables utility did not display type information for hash maps or fully replicated triggers. (Bug #24383742)
- Long message buffer exhaustion when firing immediate triggers could result in row ID leaks;
 this could later result in persistent RowId already allocated errors (NDB Error 899). (Bug #23723110)

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

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

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

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

CREATE TABLE grandchild(
  ref2 INT,
```

```
FOREIGN KEY fk2(ref2)
REFERENCES child(ref1) ON UPDATE CASCADE
) ENGINE NDB;
```

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

- The NDB binlog injector thread used an injector mutex to perform two important tasks:
 - 1. Protect against client threads creating or dropping events whenever the injector thread waited for pollEvents().
 - 2. Maintain access to data shared by the injector thread with client threads.

The first of these could hold the mutex for long periods of time (on the order of 10ms), while locking it again extremely quickly. This could keep it from obtaining the lock for data access ("starved") for unnecessarily great lengths of time.

To address these problems, the injector mutex has been refactored into two—one to handle each of the two tasks just listed.

It was also found that initialization of the binlog injector thread held the injector mutex in several places unnecessarily, when only local thread data was being initialized and sent signals with condition information when nothing being waited for was updated. These unneeded actions have been removed, along with numerous previous temporary fixes for related injector mutex starvation issues. (Bug #83676, Bug #83127, Bug #25042101, Bug #24715897)

References: See also: Bug #82680, Bug #20957068, Bug #24496910.

• 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)

• When ndbmtd was built on Solaris/SPARC with version 5.3 of the GNU tools, data nodes using the resulting binary failed during startup. (Bug #83500, Bug #24941880)

References: See also: Bug #83517, Bug #24947597.

- MySQL NDB Cluster failed to compile using GCC 6. (Bug #83308, Bug #24822203)
- 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.

Changes in MySQL NDB Cluster 7.5.4 (5.7.16-ndb-7.5.4) (2016-10-18, General Availability)

- · Functionality Added or Changed
- · Bugs Fixed

Functionality Added or Changed

• Important Change; Packaging: Naming and organization of the RPMs provided for MySQL NDB Cluster have been changed to align with those released for the MySQL server. All MySQL NDB Cluster RPM package names are now prefixed with mysql-cluster. Data nodes are now installed using the data-node package; management nodes are now installed from the management-server package; and SQL nodes require the server and common packages. Important: SQL nodes must use the mysql-cluster version of these RPMs; the versions released for the standard MySQL server do not provide support for the NDB storage engine. All client programs, including both the mysql client and the ndb_mgm management client, are now included in the client RPM.

For more information, see Installing NDB Cluster from RPM. (WL #9129)

• Important Change: Added the PARTITION_BALANCE values FOR_RA_BY_LDM_X_2, FOR_RA_BY_LDM_X_3, and FOR_RA_BY_LDM_X_4, which can be used to set the number of partitions used by each local data manager to two, three, and four partitions, respectively, in addition to FOR_RA_BY_LDM, which sets this number to one.

Use of MAX_ROWS for setting the number of partitions used by NDB tables is now deprecated and subject to removal in a future MySQL NDB Cluster version.

For more information, see Setting NDB Comment Options. (Bug #81759, Bug #23544301)

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

(Bug #22353594)

• Added the --print-sql-log option for the ndb_restore program included with the MySQL NDB Cluster distribution. This option causes the program to log SQL statements to stdout.

Note that each table being restored in this fashion must have an explicitly defined primary key; the hidden primary key implemented by the NDB storage engine is *not* sufficient for this purpose. (Bug #13511949)

- For fully replicated tables, ndb_desc shows only nodes holding main fragment replicas for partitions; nodes with copy fragment replicas only are ignored. To make this information available in the mysql client, several new tables have been introduced in the ndbinfo information database. These tables are listed here, with brief descriptions:
 - dict_obj_info provides the names and types of database (DICT) objects in NDB, such as tables and indexes, as well as information about parent objects where applicable
 - table_distribution_status provides NDB table distribution status information
 - table fragments provides information about the distribution of NDB table fragments
 - table_info provides information about logging, checkpointing, storage, and other options in force for each NDB table
 - table_replicas provides information about fragment replicas.

For more information, see ndbinfo: The NDB Cluster Information Database. (Bug #81762, Bug #23547643)

Bugs Fixed

- Important Change: The default value of the --ndb-default-column-format server option has been changed from DYNAMIC to FIXED. This has been done for backwards compatibility. Only the default has been changed; setting this option to DYNAMIC continues to cause DYNAMIC to be used for ROW FORMAT and COLUMN FORMAT unless overridden. (Bug #24487363)
- Important Change: Event buffer status reporting has been improved by altering the semantics for calculating lag or slippage. Rather than defining this lag as the number of epochs behind, lag is now taken as the number of epochs completely buffered in the event buffer, but not yet consumed by the binlog injector thread. As part of this work, the default value for the ndb_report_thresh_binlog_epoch_slip system variable has been increased from 3 to 10. For more information, see the description of this variable in the documentation, as well as Event Buffer Reporting in the Cluster Log. (Bug #22916457)

References: See also: Bug #22901309.

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

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

• NDB Cluster APIs: To process incoming signals, a thread which wants to act as a receiver must acquire polling rights from the transporter layer. This can be requested and assigned to a separate receiver thread, or each client thread can take the receiver role when it is waiting for a result.

When the thread acting as poll owner receives a sufficient amount of data, it releases locks on any other clients taken while delivering signals to them. This could make them runnable again, and the operating system scheduler could decide that it was time to wake them up, which happened at the expense of the poll owner threads, which were in turn excluded from the CPU while still holding polling rights on it. After this fix, polling rights are released by a thread before unlocking and signalling other threads. This makes polling rights available for other threads that are actively executing on this CPU.

This change increases concurrency when polling receiver data, which should also reduce latency for clients waiting to be woken up. (Bug #83129, Bug #24716756)

• NDB Cluster APIs: libndbclient and libmysqlclient exported conflicting symbols, resulting in a segmentation fault in debug builds on Linux. To fix this issue, the conflicting symbols in libndbclient.so are no longer publicly visible. Due to this change, the version number for libndbclient.so has been raised from 6.0.0 to 6.1.0. (Bug #83093, Bug #24707521)

References: See also: Bug #80352, Bug #22722555.

• NDB Cluster APIs: When NDB schema object ownership checks are enabled by a given NdbTransaction, objects used by this transaction are checked to make sure that they belong to the NdbDictionary owned by this connection. An attempt to create a NdbOperation, NdbScanOperation, or NdbIndexScanOperation on a table or index not belonging to the same connection fails.

This fix corrects a resource leak which occurred when the operation object to be created was allocated before checking schema object ownership and subsequently not released when the object creation failed. (Bug #81949, Bug #23623978)

References: See also: Bug #81945, Bug #23623251.

• NDB Cluster APIs: NDB API objects are allocated in the context of an Ndb object, or of an NdbTransaction object which is itself owned by an Ndb object. When a given Ndb object is destroyed, all remaining NdbTransaction objects are terminated, and all NDB API objects related to this Ndb object should be released at this time as well. It was found, when there remained unclosed NdbTransaction objects when their parent Ndb object was destroyed, leaks of objects allocated from the NdbTransaction objects could occur. (However, the NdbTransaction objects themselves did not leak.)

While it is advisable (and, indeed, recommended) to close an NdbTransaction explicitly as soon as its lifetime ends, the destruction of the parent Ndb object should be sufficient to release whatever objects are dependent on it. Now in cases such as described previously, the Ndb destructor checks to ensure that all objects derived from a given Ndb instance are truly released. (Bug #81945, Bug #23623251)

• NDB Cluster APIs: The term "fragment count type" has been superceded by "partition balance". This change affects NDB_TABLE options for NDB tables as well as in the NDB API. In NDB_TABLE table option syntax, the FRAGMENT_COUNT_TYPE keyword is replaced with PARTITION_BALANCE. In the NDB API, the Table methods getFragmentCountType() and setFragmentCountType() have been renamed to getPartitionBalance() and setPartitionBalance(), respectively; getFragmentCountTypeString() is renamed to getPartitionBalanceString(). In addition, Object::FragmentCountType has been renamed to PartitionBalance, and the names of its enumerated values have been updated to be consistent with the new nomenclature.

For more information on how these changes affect NDB API applications, see the indicated Table and Object member descriptions. For more information on the SQL-level changes made as part of this fix, Setting NDB Comment Options. (Bug #81761, Bug #23547525)

References: See also: Bug #83147, Bug #24733331.

• NDB Cluster APIs: In some of the NDB API example programs included with the MySQL NDB Cluster distribution, ndb_end() was called prior to calling the Ndb_cluster_connection destructor. This caused a segmentation fault in debug builds on all platforms. The example programs affected have also been extensively revised and refactored. See NDB API Examples, for more information. (Bug #80352, Bug #22722555)

References: See also: Bug #83093, Bug #24707521.

- If more than 4096 seconds elapsed while calculating an internal NdbDuration::microSec() value, this could cause an assert warning that the calculation would overflow. We fix this to avoid any overflow or precision loss when converting from the internal "tick" format to microseconds and nanoseconds, by performing the calculation in two parts corresponding to seconds and fractions of a second. (Bug #24695026)
- The serial commit protocol—which commits each operation at each replica individually and serially, and is used by the DBTC kernel block (see The DBTC Block) for takeover and when a transaction is judged to have timed out during the COMMIT or COMPLETE phase—had no support for LATE_COMMIT, which is required for the READ_BACKUP and FULLY_REPLICATED protocols. (Bug #24681305)
- In some cases, ALTER TABLE ... REORGANIZE PARTITION could lead to an unplanned shutdown of the cluster. This was due to the fact that, for fully replicated tables. the log part ID was assumed to be the same as the partition ID. This worked when FOR_RA_BY_LDM was used, but not necessarily for the other partition balancing types. (Bug #24610551)
- Using ALGORITHM=INPLACE when changing any of a table's NDB_TABLE properties (see Setting NDB Comment Options) caused the server to fail. (Bug #24584741)
- Following successive ALTER TABLE statements updating NDB_TABLE properties (see Setting NDB Comment Options), the current values were not always shown by SHOW CREATE TABLE or ndb_desc. (Bug #24584690)
- Case-insensitivity of keywords such as FULLY_REPLICATED in NDB_TABLE comments was not honored. (Bug #24577931)
- An ALTER TABLE statement attempting to set both FULLY_REPLICATED and PARTITION_BALANCE (see Setting NDB Comment Options) failed with a garbled error message. (Bug #24577894)
- A number of dependencies between the binlog injector thread and the NDB utility thread—a recurring source of synchronization and other problems—were removed. The principal changes are listed here:
 - Moved the setup of binlog injector structures from the utility thread to the injector thread itself.
 - Removed sharing of some utility and injector thread structures between these threads.
 - Moved stopping of the utility thread from the injector thread into a common block in which other such threads are stopped.
 - · Removed a number of hacks required by the previous design.
 - Removed some injector mutex locking and injector condition signaling which were made obsolete by the changes already listed.

(Bug #24496910)

References: See also: Bug #22204186.

 A late commit ACK signal used for FULLY_REPLICATED or READ_BACKUP tables caused the associated ApiConnectionRecord to have an invalid state. (Bug #24459817) References: See also: Bug #24444861.

- Added missing error information for a failure occurring when tables on disk became full. (Bug #24425373)
- When ndbmtd crashed, the resulting error log incorrectly specified the name of the trace for thread
 0, appending the nonexistent suffix to to the file name. (Bug #24353408)
- Passing a nonexistent node ID to CREATE NODEGROUP led to random data node failures. (Bug #23748958)
- DROP TABLE followed by a node shutdown and subesequent master takeover—and with the
 containing local checkpoint not yet complete prior to the takeover—caused the LCP to be ignored,
 and in some cases, the data node to fail. (Bug #23735996)

References: See also: Bug #23288252.

- Removed an invalid assertion to the effect that all cascading child scans are closed at the time
 API connection records are released following an abort of the main transaction. The assertion was
 invalid because closing of scans in such cases is by design asynchronous with respect to the main
 transaction, which means that subscans may well take some time to close after the main transaction
 is closed. (Bug #23709284)
- Although arguments to the DUMP command are 32-bit integers, ndb_mgmd used a buffer of only 10 bytes when processing them. (Bug #23708039)
- The READ_BACKUP setting was not honored when performing scans on BLOB tables. (Bug #23703536)
- Setting FULLY_REPLICATED=1 (see Setting NDB Comment Options) did not propagate to the internal BLOB part tables used for BLOB and TEXT columns. (Bug #23703343)
- The READ BACKUP setting was not applied to unique indexes. (Bug #23702848)
- In ReadCommitted mode, DBSPJ read primary fragment replicas for tables with READ_BACKUP (see Setting NDB Comment Options), even when a local fragment was available. (Bug #23633848)
- ALL REPORT MemoryUsage produced incorrect output when fully replicated tables were in use. (Bug #23539805)
- Ordered indexes did not inherit READ_BACKUP (see Setting NDB Comment Options) from an indexed table, which meant that ordered index scans continued to be routed to only to primary fragment replicas and never to backup fragment replicas.

Now DBDICT sets this property on ordered indexes from the table property when it distributes this information to instances of DBTC and DBSPJ. (Bug #23522027)

- Updates to a table containing a virtual column could cause the binary logging thread to fail. (Bug #23514050)
- A number of potential buffer overflow issues were found and fixed in the NDB codebase. (Bug #23152979)
- During an online upgrade from a MySQL NDB Cluster 7.3 release to an NDB 7.4 (or later) release, the failures of several data nodes running the lower version during local checkpoints (LCPs), and just prior to upgrading these nodes, led to additional node failures following the upgrade. This was due to lingering elements of the EMPTY_LCP protocol initiated by the older nodes as part of an LCP-plus-restart sequence, and which is no longer used in NDB 7.4 and later due to LCP optimizations implemented in those versions. (Bug #23129433)
- A SIGNAL_DROPPED_REP handler invoked in response to long message buffer exhaustion
 was defined in the SPJ kernel block, but not actually used. This meant that the default handler

from SimulatedBlock was used instead in such cases, which shut down the data node. (Bug #23048816)

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

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

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

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

- ndb_restore set the MAX_ROWS attribute for a table for which it had not been set prior to taking the backup. (Bug #22904640)
- Whenever data nodes are added to or dropped from the cluster, the NDB kernel's Event API is notified of this using a SUB_GCP_COMPLETE_REP signal with either the ADD (add) flag or SUB (drop) flag set, as well as the number of nodes to add or drop; this allows NDB to maintain a correct count of SUB_GCP_COMPLETE_REP signals pending for every incomplete bucket. In addition to handling the bucket for the epoch associated with the addition or removal, it must also compensate for any later incomplete buckets associated with later epochs. Although it was possible to complete such buckets out of order, there was no handling of these, leading a stall in to event reception.

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

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

- When performing online reorganization of tables, unique indexes were not included in the reorganization. (Bug #13714258)
- Under very high loads, neither the receive thread nor any user thread had sufficient capacity to handle poll ownership properly. This meant that, as the load and the number of active thread increased, it became more difficult to sustain throughput. This fix attempts to increase the priority of the receive thread and retains poll ownership if successful.

This fix requires sufficient permissions to be enabled. On Linux systems, this means ensuring that either the data node binary or the user it runs as has permission to change the nice level. (Bug #83217, Bug #24761073)

- When restoring a backup taken from a database containing tables that had foreign keys,
 ndb_restore disabled the foreign keys for data, but not for the logs. (Bug #83155, Bug #24736950)
- Local reads of unique index and blob tables did not work correctly for fully replicated tables using more than one node group. (Bug #83016, Bug #24675602)
- The effects of an ALTER TABLE statement changing a table to use READ_BACKUP were not preserved after a restart of the cluster. (Bug #82812, Bug #24570439)
- Using FOR_RP_BY_NODE or FOR_RP_BY_LDM for PARTITION_BALANCE did not work with fully replicated tables. (Bug #82801, Bug #24565265)
- Changes to READ_BACKUP settings were not propagated to internal blob tables. (Bug #82788, Bug #24558232)

- The count displayed by the c_exec column in the ndbinfo.threadstat table was incomplete. (Bug #82635, Bug #24482218)
- The default PARTITION_BALANCE setting for NDB tables created with READ_BACKUP=1 (see Setting NDB Comment Options) has been changed from FOR_RA_BY_LDM to FOR_RP_BY_LDM. (Bug #82634, Bug #24482114)
- The internal function ndbcluster_binlog_wait(), which provides a way to make sure that all events originating from a given thread arrive in the binary log, is used by SHOW BINLOG EVENTS as well as when resetting the binary log. This function waits on an injector condition while the latest global epoch handled by NDB is more recent than the epoch last committed in this session, which implies that this condition must be signalled whenever the binary log thread completes and updates a new latest global epoch. Inspection of the code revealed that this condition signalling was missing, and that, instead of being awakened whenever a new latest global epoch completes (~100ms), client threads waited for the maximum timeout (1 second).

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

- Fully replicated internal foreign key or unique index triggers could fire multiple times, which led to
 aborted transactions for an insert or a delete operation. This happened due to redundant deferred
 constraint triggers firing during pre-commit. Now in such cases, we ensure that only triggers specific
 to unique indexes are fired in this stage. (Bug #82570, Bug #24454378)
- Backups potentially could fail when using fully replicated tables due to their high usage (and subsequent exhaustion) of internal trigger resources. To compensate for this, the amount of memory reserved in the NDB kernel for internal triggers has been increased, and is now based in part on the maximum number of tables. (Bug #82569, Bug #24454262)

References: See also: Bug #23539733.

In the DBTC function executeFullyReplicatedTrigger() in the NDB kernel, an incorrect check
of state led in some cases to failure handling when no failure had actually occurred. (Bug #82568,
Bug #24454093)

References: See also: Bug #23539733.

When returning from LQHKEYREQ with failure in LQHKEYREF in an internal trigger operation, no
check was made as to whether the trigger was fully replicated, so that those triggers that were fully
replicated were never handled. (Bug #82566, Bug #24453949)

References: See also: Bug #23539733.

- When READ_BACKUP had not previously been set, then was set to 1 as part of an ALTER
 TABLE ... ALGORITHM=INPLACE statement, the change was not propagated to internal unique
 index tables or BLOB tables. (Bug #82491, Bug #24424459)
- Distribution of MySQL privileges was incomplete due to the failure of the mysql_cluster_move_privileges() procedure to convert the mysql.proxies_priv table to NDB. The root cause of this was an ALTER TABLE ... ENGINE NDB statement which sometimes failed when this table contained illegal TIMESTAMP values. (Bug #82464, Bug #24430209)
- The internal variable m_max_warning_level was not initialized in storage/ndb/src/kernel/blocks/thrman.cpp. This sometimes led to node failures during a restart when the uninitialized value was treated as 0. (Bug #82053, Bug #23717703)
- Usually, when performing a system restart, all nodes are restored from redo logs and local
 checkpoints (LCPs), but in some cases some node might require a copy phase before it is finished
 with the system restart. When this happens, the node in question waits for all other nodes to start
 up completely before performing the copy phase. Notwithstanding the fact that it is thus possible to
 begin a local checkpoint before reaching start phase 4 in the DBDIH block, LCP status was initialized

to IDLE in all cases, which could lead to a node failure. Now, when performing this variant of a system restart, the LCP status is no longer initialized. (Bug #82050, Bug #23717479)

• After adding a new node group online and executing ALTER TABLE ... ALGORITHM=INPLACE REORGANIZE PARTITION, partition IDs were not set correctly for new fragments.

In a related change done as part of fixing this issue, ndb_desc -p now displays rows relating to partitions in order of partition ID. (Bug #82037, Bug #23710999)

- When executing STOP BACKUP it is possible sometimes that a few bytes are written to the backup
 data file before the backup process actually terminates. When using ODIRECT, this resulted in the
 wrong error code being returned. Now in such cases, nothing is written to O_DIRECT files unless the
 alignment is correct. (Bug #82017, Bug #23701911)
- When transaction coordinator (TC) connection records were used up, it was possible to handle scans only for local checkpoints and backups, so that operations coming from the DBUTIL block—used for ALTER TABLE ... REORGANIZE PARTITION and other operations that reorganize metadata—were unnecessarily blocked. In addition, such operations were not always retried when TC records were exhausted. To fix this issue, a number of operation records are now earmarked for DBUTIL usage, as well as for LCP and backup usage so that these operations are also not negatively impacted by operations coming from DBUTIL.

For more information, see The DBUTIL Block. (Bug #81992, Bug #23642198)

- Operations performing multiple updates of the same row within the same transaction could sometimes lead to corruption of lengths of page entries. (Bug #81938, Bug #23619031)
- During a node restart, a fragment can be restored using information obtained from local checkpoints (LCPs); up to 2 restorable LCPs are retained at any given time. When an LCP is reported to the DIH kernel block as completed, but the node fails before the last global checkpoint index written into this LCP has actually completed, the latest LCP is not restorable. Although it should be possible to use the older LCP, it was instead assumed that no LCP existed for the fragment, which slowed the restart process. Now in such cases, the older, restorable LCP is used, which should help decrease long node restart times. (Bug #81894, Bug #23602217)
- Optimized node selection (ndb_optimized_node_selection setting) was not respected by ndb_data_node_neighbour when this was enabled. (Bug #81778, Bug #23555834)
- NDB no longer retries a global schema lock if this has failed due to a timeout (default 3000ms) and there is the potential for this lock request to participate in a metadata lock-global schema lock deadlock. Now in such cases it selects itself as a "victim", and returns the decision to the requestor of the metadata lock, which then handles the request as a failed lock request (preferable to remaining deadlocked indefinitely), or, where a deadlock handler exists, retries the metadata lock-global schema lock. (Bug #81775, Bug #23553267)
- Two issues were found in the implementation of hash maps—used by NDB for mapping a table row's hash value to a partition—for fully replicated tables:
 - 1. Hash maps were selected based on the number of fragments rather than the number of partitions. This was previously undetected due to the fact that, for other kinds of tables, these values are always the same.
 - 2. The hash map was employed as a partition-to-partition map, using the table row's hash value modulus the partition count as input.

This fix addresses both of the problems just described. (Bug #81757, Bug #23544220)

References: See also: Bug #81761, Bug #23547525, Bug #23553996.

• Using mysqld together with --initialize and --ndbcluster led to problems later when attempting to use mysql_upgrade. When running with --initialize, the server does not require

NDB support, and having it enabled can lead to issues with ndbinfo tables. To prevent this from happening, using the --initialize option now causes mysqld to ignore the --ndbcluster option if the latter is also specified.

This issue affects upgrades from MySQL NDB Cluster 7.5.2 or 7.5.3 only. In cases where such upgrades fail for the reasons outlined previously, you can work around the issue by deleting all .frm files in the data/ndbinfo directory following a rolling restart of the entire cluster, then running mysql_upgrade. (Bug #81689, Bug #23518923)

References: See also: Bug #82724, Bug #24521927.

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

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

- ALTER TABLE ... ALGORITHM=INPLACE on a fully replicated table did not copy the associated trigger ID, leading to a failure in the DBDICT kernel block. (Bug #81544, Bug #23330359)
- The list of deferred tree node lookup requests created when preparing to abort a DBSPJ request were not cleared when this was complete, which could lead to deferred operations being started even after the DBSPJ request aborted. (Bug #81355, Bug #23251423)

References: See also: Bug #23048816.

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

References: See also: Bug #23048816.

- The read backup feature added in MySQL NDB Cluster 7.5.2 that makes it possible to read from backup fragment replicas was not used for reads with lock, or for reads of BLOB tables or unique key tables where locks were upgraded to reads with lock. Now the TCKEYREQ and SCAN_TABREQ signals use a flag to convey information about such locks making it possible to read from a backup fragment replica when a read lock was upgraded due to being the read of the base table for a BLOB table, or due to being the read for a unique key. (Bug #80861, Bug #23001841)
- Primary fragment replicas of partitioned tables were not distributed evenly among node groups and local data managers.

As part of the fix for this issue, the maximum number of node groups supported for a single MySQL NDB Cluster, which was previously not determined, is now set at 48 (MAX_NDB_NODE_GROUPS). (Bug #80845, Bug #22996305)

- Several object constructors and similar functions in the NDB codebase did not always perform sanity checks when creating new instances. These checks are now performed under such circumstances. (Bug #77408, Bug #21286722)
- An internal call to malloc() was not checked for NULL. The function call was replaced with a direct write. (Bug #77375, Bug #21271194)

Changes in MySQL NDB Cluster 7.5.3 (5.7.13-ndb-7.5.3) (2016-07-08, Release Candidate)

- · Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

Important Change: It is now possible to set READ_BACKUP for an existing table
 online using an SQL statement such as ALTER TABLE . . . ALGORITHM=INPLACE,
 COMMENT="NDB_TABLE=READ_BACKUP=1". See Setting NDB Comment Options, for further
 information about the READ_BACKUP option. (Bug #80858, Bug #23001617)

References: See also: Bug #18435416.

- Added three new tables to the ndbinfo information database to provide running information about locks and lock attempts in an active MySQL NDB Cluster. These tables, with brief descriptions, are listed here:
 - cluster_locks: Current lock requests which are waiting for or holding locks; this information can be useful when investigating stalls and deadlocks. Analogous to cluster operations.
 - locks_per_fragment: Counts of lock claim requests, and their outcomes per fragment, as well as total time spent waiting for locks successfully and unsuccessfully. Analogous to operations_per_fragment and memory_per_fragment.
 - server_locks: Subset of cluster transactions—those running on the local mysqld, showing a connection ID per transaction. Analogous to server_operations.

For more information, see ndbinfo: The NDB Cluster Information Database. (WL #7656)

 The NDB storage engine now supports generated columns (see CREATE TABLE and Generated Columns) as well as indexes on stored generated columns.

Indexes on virtual generated columns of NDB tables are not supported.

ALTER TABLE statements that add stored generated columns to NDB tables cannot be performed online. (WL #8188, WL #9048)

Bugs Fixed

- NDB Cluster APIs: The scan lock takeover issued by NdbScanOperation::lockCurrentTuple() did not set the operation type for the takeover operation. (Bug #23314028)
- The ndbinfo cpustat_1sec and cpustat_20sec tables did not provide any history information. (Bug #23520271)
- During shutdown, the mysqld process could sometimes hang after logging NDB Util: Stop ... NDB Util: Wakeup. (Bug #23343739)

References: See also: Bug #21098142.

 During expansion or reduction of a hash table, allocating a new overflow page in the DBACC kernel block caused the data node to fail when it was out of index memory. This could sometimes occur after a large table had increased or decreased very rapidly in size. (Bug #23304519)

References: This issue is a regression of: Bug #13436216.

 When tracking time elapsed from sending a SCAN_FRAGREQ signal to receiving a corresponding SCAN_FRAGCONF, the assumption was made in the DBTC kernel block that a SCAN_FRAGCONF can occur only after sending a SCAN_FRAGREQ or SCAN_NEXTREQ signal, which is not always the case: It is actually possible that a local query handler can, immediately after sending a SCAN_FRAGCONF, send an additional SCAN_FRAGCONF signal upon reporting that the scan is closed. This problem is fixed by ensuring that the timer value is initialized each time before use. (Bug #81907, Bug #23605817, Bug #23306695)

- Table indexes were listed in the output of ndb_desc in a nondeterministic order that could vary between platforms. Now these indexes are ordered by ID in the output. (Bug #81763, Bug #23547742)
- Following a restart of the cluster, the first attempt to read from any of the ndbinfo cpustat, cpustat_50ms, cpustat_1sec, or cpustat_20sec tables generated a warning to the effect that columns were missing from the table. Subsequently, the thread_sleeping and spin_time columns were found to be missing from each of these tables. (Bug #81681, Bug #23514557)

References: See also: Bug #23305078.

• Using a ThreadConfig parameter value with a trailing comma led to an assertion. (Bug #81588, Bug #23344374)

Changes in MySQL NDB Cluster 7.5.2 (5.7.12-ndb-7.5.2) (2016-06-01, Development Milestone)

- · Functionality Added or Changed
- · Bugs Fixed

Functionality Added or Changed

• Important Change; NDB Replication: The ndb_binlog_index table now uses the InnoDB storage engine. When upgrading previous versions to the current release or to a later one, use mysql_upgrade with --force --upgrade-system-tables to have it perform ALTER TABLE ... ENGINE=INNODB on this table. Use of the MyISAM storage engine for this table continues to be supported for backward compatibility.

One benefit of this change is that it is now possible to depend on transactional behavior and lock-free reads for this table, which should help alleviate concurrency issues during purge operations and log rotation, and improve the availability of this table.

Due to differences in storage requirements for MyISAM and InnoDB tables, users should be aware that ndb_binlog_index may now take up more disk space than was previously required.

For more information, see NDB Cluster Replication Schema and Tables. (WL #7162)

• **Performance:** A deficiency in event buffer memory allocation was identified as inefficient and possibly leading to undesirable results. This could happen when additional memory was allocated from the operating system to buffer received event data even when memory had already been allocated but remained unused. This is fixed by allocating the event buffer memory directly from the page allocation memory manager (mmap()), where such functionality is offered by the operating system, allowing for direct control over the memory such that it is in fact returned to the system when released.

This remimplementation avoids the tendencies of the existing one to approach worst-case memory usage, maintainence of data structures for a worst-case event buffer event count, and useless caching of free memory in unusable positions. This work should also help minimize the runtime costs of buffering events, minimize heap fragmentation, and avoid OS-specific problems due to excessive numbers of distinct memory mappings.

In addition, the relationship between epochs and internal EventData objects is now preserved throughout the event lifecycle, reception to consumption, thus removing the need for iterating, and keeping in synch, two different lists representing the epochs and their EventData objects.

As part of this work, better reporting on the relevant event buffer metrics is now provided in the cluster logs. (WL #7677, WL #9207)

References: See also: Bug #21651536, Bug #21660947, Bug #21661297, Bug #21673318, Bug #21689380, Bug #21809959.

- NDB Cluster APIs: Added the Ndb::setEventBufferQueueEmptyEpoch() method, which makes it possible to enable queuing of empty events (event type TE_EMPTY). (Bug #22157845)
- NDB Cluster APIs: Ndb_free_list_t is a template used in the implementation of the NDB API to create a list keeping released API objects such as NdbTransaction, NdbOperation, and NdbRecAttr. One drawback to this template is that released API objects are kept in the list for the lifetime of the owning Ndb object, such that a transient peak in the demand for any object causes an effective leak in memory that persists until this Ndb object itself has been released.

This work adds statistics to each Ndb_free_list instance which samples the usage of objects maintained by the list; now, when objects are released, they can be released into the free list, or deallocated, based on the collected usage statistics. (WL #8351)

- **JSON:** The NDB storage engine supports the MySQL JSON data type and MySQL JSON functions implemented in MySQL 5.7.8 and later. This support is subject to the limitation that a single NDB table can have at most 3 JSON columns. (WL #9007)
- MySQL NDB ClusterJ: To make it easier for ClusterJ to handle fatal errors that require the SessionFactory to be closed, a new public method in the SessionFactory interface, getConnectionPoolSessionCounts(), has been created. When it returns zeros for all pooled connections, it means all sessions have been closed, at which point the SessionFactory can be closed and reopened. See Error Handling and Reconnection for more detail. (Bug #22353594)
- Made the following enhancements and additions to the ThreadConfig multithreaded data node (ndbmtd) configuration parameter:
 - Added support for non-exclusive CPU locking on FreeBSD and Windows using cpubind and cpuset.
 - Added support for exclusive CPU locking on Solaris, using cpubind_exclusive and cpuset_exclusive, which are added in this release.
 - Added thread prioritization using thread_prio, which is added in this release. thread_prio is supported on Linux, FreeBSD, Windows, and Solaris, but the exact effects of this setting are platform-specific; see the documentation for details.
 - Added support for setting realtime on Windows platforms.

For more information, see the description of the ThreadConfig parameter in the online documentation. (Bug #25830247, WL #9096)

- ndb_restore now performs output logging for specific stages of its operation. (Bug #21097957)
- An improvement in the hash index implementation used by MySQL NDB Cluster data nodes means that partitions may now contain more than 16 GB of data for fixed columns, and the maximum partition size for fixed column data is now increased to 128 TB. The previous limitation originated with the DBACC block in the NDB kernel using only 32-bit references to the fixed-size part of a row handled in the DBTUP block, even though 45-bit references were already in use elsewhere in the kernel outside the DBACC block; all such references in DBACC now use 45-bit pointers instead.

As part of this work, error messages returned by the DBACC kernel block that were overly generic have now been improved by making them more specific. (Bug #13844581, Bug #17465232, WL #8570, WL #8817, WL #8962)

- A number of changes and improvements were made to the handling of send threads by NDB. The changes are summarized with brief descriptions in the following list:
 - Decreased resource requirements for send threads, making sure that a given configuration using send threads outperforms the same configuration without send threads.
 - Made use of otherwise idle threads (other than receiver threads) as send threads without incurring extra CPU resources in low-load situations.
 - Improved response time for write transactions.
 - Provided for handling of bad configuration data in a more graceful manner.
 - Made it possible to measure CPU usage with improved real-time reporting from data nodes on their resource usage.

As part of implementing the last item of those just listed, a number of new tables providing information about CPU and thread activity by node, thread ID, and thread type have been added to the ndbinfo information database. These tables are listed here:

- cpustat: Provides per-second, per-thread CPU statistics
- cpustat_50ms: Shows raw per-thread CPU statistics data, gathered every 50ms
- cpustat_1sec: Provides raw per-thread CPU statistics data, gathered each second
- cpustat_20sec: Displays raw per-thread CPU statistics data, gathered every 20 seconds
- threads: Shows names and descriptions of thread types

For more information, see ndbinfo: The NDB Cluster Information Database. (WL #8968)

- Rewrote the implementation of the NDB storage engine's global schema lock functionality to make use of the metadata lock hooks implemented in the MySQL Server in MySQL 5.7.11. (WL #8331)
- A number of improvements provide additional read scalability for NDB by making it possible to
 read tables locally. It is now possible to enable reads from any fragment replica, rather than from
 the primary fragment replica only. This is disabled by default to remain compatible with previous
 behavior, but can be enabled for a given SQL node using the ndb_read_backup system variable
 added in this release.

It also becomes possible to be more flexible about the assignment of partitions by setting a fragment count type. Possible count types are one per node, one per node group, one per Local Data Manager (LDM) per node (this is the previous assignment scheme and still the default), and one per LDM per node group. This setting can be controlled for individual tables by means of a FRAGMENT_COUNT_TYPE option embedded in an NDB_TABLE comment in CREATE TABLE or ALTER TABLE.

This also means that, when restoring table schemas, ndb_restore --restore-meta now uses the default partitioning for the target cluster, rather than duplicating the partitioning of the cluster from which the backup was taken. This is useful when restoring to a cluster having more data nodes than the original. See Restoring to More Nodes Than the Original, for more information.

Tables using one of the two per-node-group settings for the fragment count type can also be fully replicated. This requires that the table's fragment count type is <code>ONE_PER_NODE_GROUP</code> or <code>ONE_PER_LDM_PER_NODE_GROUP</code>, and can be enabled using the option <code>FULLY_REPLICATED=1</code> within in an <code>NDB_TABLE</code> comment. The option can be enabled by default for all new <code>NDB</code> tables using the <code>ndb_fully_replicated</code> system variable added in this release.

Settings for table-level READ_BACKUP are also supported using the COMMENT="NDB_TABLE=..." syntax. It is also possible (and often preferable) to set multiple options in one comment within a

single CREATE TABLE or ALTER TABLE statement. For more information and examples, see Setting NDB Comment Options.

This release also introduces the ndb_data_node_neighbour system variable, which is intended to be employed with transaction hinting (and fully-replicated tables), and to provide the current SQL node with the ID of a "nearby" data node to use. See the description of this variable in the documentation for more information. (WL #9018, WL #9019, WL #9020)

References: See also: Bug #18435416, Bug #11762155, Bug #54717.

Bugs Fixed

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

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

- Important Change: When started with the --initialize option, mysqld no longer enables the NDBCLUSTER storage engine plugin. This change was needed to prevent attempted initialization of system databases as distributed (rather than as specific to individual SQL nodes), which could result in a metadata lock deadlock. This fix also brings the behavior of --initialize in this regard into line with that of the discontinued --bootstrap option, which started a minimal mysqld instance without enabling NDB. (Bug #22758238)
- **Performance:** A performance problem was found in an internal polling method do_poll() where the polling client did not check whether it had itself been woken up before completing the poll. Subsequent analysis showed that it is sufficient that only some clients in the polling queue receive data. do_poll() can then signal these clients and give up its polling rights, even if the maximum specified wait time (10 ms) has not expired.

This change allows do_poll() to continue polling until either the maximum specified wait time has expired, or the polling client itself has been woken up (by receiving what it was waiting for). This avoids unnecessary thread switches between client threads and thus reduces the associated overhead by as much as 10% in the API client, resulting in a significant performance improvement when client threads perform their own polling. (Bug #81229, Bug #23202735)

- macOS: On OS X, ndb_config failed when an empty string was used for the --host option. (Bug #80689, Bug #22908696)
- Microsoft Windows: ndb_mgmd failed to start on 32-bit Windows platforms, due to an issue with calling dynamically loaded functions; such issues were also likely to occur with other NDB programs using ndb_init(). It was found that all of the functions used are already supported in targeted versions of Windows, so this problem is fixed by removing the dynamic loading of these functions and using the versions provided by the Windows header files instead. (Bug #80876, Bug #23014820)
- Microsoft Windows: When building MySQL NDB Cluster on Windows using more than one parallel build job it was sometimes possible for the build to fail because host_info.exe could not be installed. To fix this problem, the install_mcc target is now always built prior to the host_info target. (Bug #80051, Bug #22566368)
- **Microsoft Windows:** Performing ANALYZE TABLE on a table having one or more indexes caused ndbmtd to fail with an *InvalidAttrInfo* error due to signal corruption. This issue occurred consistently on Windows, but could also be encountered on other platforms. (Bug #77716, Bug #21441297)

- **Solaris:** The ndb_print_file utility failed consistently on Solaris 9 for SPARC. (Bug #80096, Bug #22579581)
- NDB Disk Data: The following improvements were made to logging during restarts by data nodes using MySQL NDB Cluster Disk Data:
 - The total amount of undo log to be applied by the data node is now provided as the total number of pages present in the log. This is a worst case estimate.
 - Progress information is now provided at regular intervals (once for each 30000 records) as the
 undo log is applied. This information is supplied as the number of records and number of undo log
 pages applied so far during the current restart.

(Bug #22513381)

- NDB Cluster APIs: Deletion of Ndb objects used a dispoportionately high amount of CPU. (Bug #22986823)
- NDB Cluster APIs: Executing a transaction with an NdbIndexOperation based on an obsolete unique index caused the data node process to fail. Now the index is checked in such cases, and if it cannot be used the transaction fails with an appropriate error. (Bug #79494, Bug #22299443)
- Reserved send buffer for the loopback transporter, introduced in MySQL NDB Cluster 7.4.8 and used by API and management nodes for administrative signals, was calculated incorrectly. (Bug #23093656, Bug #22016081)

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

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

References: See also: Bug #23221573.

- Following the forced shutdown of one of the 2 data nodes in a cluster where NoOfReplicas=2, the
 other data node shut down as well, due to arbitration failure. (Bug #23006431)
- Aborting a CREATE LOGFILE GROUP statement which had failed due to lack of shared global memory was not performed correctly, causing node failure. In addition, the transaction in which this occurred was not rolled back correctly, also causing any subsequent CREATE LOGFILE GROUP to fail. (Bug #22982618)
- The ndbinfo.tc_time_track_stats table uses histogram buckets to give a sense of the distribution of latencies. The sizes of these buckets were also reported as HISTOGRAM BOUNDARY INFO messages during data node startup; this printout was redundant and so has been removed. (Bug #22819868)
- Online upgrades from previous versions of MySQL NDB Cluster to MySQL NDB Cluster 7.5 were not
 possible due to missing entries in the matrix used to test upgrade compatibility between versions.
 (Bug #22024947)
- A failure occurred in DBTUP in debug builds when variable-sized pages for a fragment totalled more than 4 GB. (Bug #21313546)
- Restoration of metadata with ndb_restore -m occasionally failed with the error message Failed to create index... when creating a unique index. While disgnosing this problem, it was found that the internal error PREPARE_SEIZE_ERROR (a temporary error) was reported as an unknown error. Now in such cases, ndb_restore retries the creation of the unique index, and PREPARE_SEIZE_ERROR is reported as NDB Error 748 Busy during read of event table. (Bug #21178339)

References: See also: Bug #22989944.

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

References: See also: Bug #23343739.

- The following improvements were made to the data node error logging mechanism:
 - Increased the message slot size 499 bytes to 999 bytes to prevent log messages from overwriting one another or from being truncated.
 - Added a Trace file name field to the output. This field contains the trace file name (without any path) and trace file number for the thread causing the trace.
 - ndbmtd trace files are also now shown in the error log.

(Bug #21082710)

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

See also The DBDICT Block. (Bug #20368450)

References: See also: Bug #20368354.

- When a write to the ndb_binlog_index table failed during a MySQL Server shutdown, mysqld killed the NDB binary logging thread. (Bug #81166, Bug #23142945)
- Memory associated with table descriptions was not freed by the internal table information method NdbDictInterface::parseTableInfo(). (Bug #81141, Bug #23130084)
- Improved memory usage by the internal TransporterFacade constructor when performing mutex array initialization. (Bug #81134, Bug #23127824)
- Fixed a memory leak that occurred when an error was raised in ha_ndbcluster::get_metadata() or one of the functions which this method calls. (Bug #81045, Bug #23089566)
- An internal function used to validate connections failed to update the connection count when creating
 a new Ndb object. This had the potential to create a new Ndb object for every operation validating
 the connection, which could have an impact on performance, particularly when performing schema
 operations. (Bug #80750, Bug #22932982)
- A table scan on an NDB table using neither an ordered index nor any Disk Data columns normally
 uses an ACC scan. If this happened while scanning an unique but unordered index which shrank (due
 to rows being deleted) after the scan started and then grew again (rows inserted), a single row that
 had been neither deleted nor inserted could be scanned twice. (Bug #80733, Bug #22926938)
- Starting a backup in the ndb_mgm client after creating a large number of tables caused a forced shutdown of the cluster. (Bug #80640, Bug #228849958)
- When an SQL node was started, and joined the schema distribution protocol, another SQL node, already waiting for a schema change to be distributed, timed out during that wait. This was because

the code incorrectly assumed that the new SQL node would also acknowledge the schema distribution even though the new node joined too late to be a participant in it.

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

- The MySQL NDB Cluster Auto-Installer failed to work in various ways on different platforms. (Bug #79853, Bug #22502247)
- The internal function ndbrequire(), which, like assert(), evaluates a given expression and terminates the process if the expression does not evaluate as true, now includes the failed expression in its output to the error logs. (Bug #77021, Bug #21128318)
- Trying to drop a table during an ongoing backup failed with the error message Unknown table;
 now, it fails with Unable to alter table as backup is in progress. (Bug #47301, Bug #11755512)

References: See also: Bug #44695, Bug #11753280.

Changes in MySQL NDB Cluster 7.5.1 (5.7.11-ndb-7.5.1) (2016-03-18, Development Milestone)

- · Functionality Added or Changed
- · Bugs Fixed

Functionality Added or Changed

• Important Change: When creating or adding columns to NDB tables, the default value used for both the COLUMN_FORMAT and ROW_FORMAT options is now DYNAMIC rather than FIXED.

This change does not affect the row format or column format used by existing tables. New columns added to such tables use the new defaults, and existing columns are changed to use these as well, provided that the ALTER TABLE statement in question uses ALGORITHM=COPY. Note that this cannot be done implicitly if mysqld is run with --ndb-allow-copying-alter-table=FALSE (the default is TRUE).

A new MySQL server option --ndb-default-column-format is added for backwards compatibility; set this to FIXED to force the old defaults to be used for COLUMN_FORMAT and ROW FORMAT.



Note

This behavior change is superseded in MySQL NDB Cluster 7.5.4.

(WL #8573)

• Scans have been improved by replacing the DIH_SCAN_GET_NODES_REQ signal, formerly used for communication between the DBTC and DBDIH kernel blocks in NDB, with the DIGETNODESREQ signal, which supports direct execution and allows for the elimination of DIH_SCAN_GET_NODES_REF and DIH_SCAN_GET_NODES_CONF, as well as for DIH_SCAN_TAB_REQ and DIH_SCAN_TAB_COMPLETE_REP signals to EXECUTE_DIRECT. This enables enable higher scalability of data nodes when used for scan operations by decreasing the use of CPU resources for scan operations by an estimated five percent in some cases. This should also improve response times, which could help prevent issues with overload of the main threads.

As part of these changes, scans made in the BACKUP kernel block have also been improved and made more efficient. (WL #8937)

References: See also: Bug #80640, Bug #22884995.

- The following improvements have been made to event buffer reporting in the cluster log:
 - Each report now identifies the API node that sent it.
 - The fields shown in the report have been improved and expanded. Percentages are now better specified and used only when appropriate. For improved clarity, the apply_epoch and latest_epoch fields have been renamed to latest_consumed_epoch and latest_buffered_epoch, respectively. The ID of the Ndb object serving as the source of the report is now shown, as is the reason for making the report (as the report reason field).
 - The frequency of unnecessary reporting has been reduced by limiting reports to those showing only significant changes in event buffer usage.
 - The MGM API adds a new NDB_LE_EventBufferStatus2 event type to handle the additional information provided by the new event buffer reporting. The NDB_LE_EventBufferStatus event type used in older versions of MySQL NDB Cluster is now deprecated, and will eventually be removed.

For more information, see Event Buffer Reporting in the Cluster Log, as well as The ndb_logevent Structure. (WL #8626)

Bugs Fixed

- Important Change: The minimum value for the BackupDataBufferSize data node configuration parameter has been lowered from 2 MB to 512 KB. The default and maximum values for this parameter remain unchanged. (Bug #22749509)
- **OS X:** Processing of local checkpoints was not handled correctly on Mac OS X, due to an uninitialized variable. (Bug #80236, Bug #22647462)
- Microsoft Windows: Compilation of MySQL with Visual Studio 2015 failed in ConfigInfo.cpp, due to a change in Visual Studio's handling of spaces and concatenation. (Bug #22558836, Bug #80024)
- Microsoft Windows: When setting up event logging for ndb_mgmd on Windows, MySQL NDB Cluster tries to add a registry key to HKEY_LOCAL_MACHINE, which fails if the user does not have access to the registry. In such cases ndb_mgmd logged the error Could neither create or open key, which is not accurate and which can cause confusion for users who may not realize that file logging is available and being used. Now in such cases, ndb_mgmd logs a warning Could not create or access the registry key needed for the application to log to the Windows EventLog. Run the application with sufficient privileges once to create the key, or add the key manually, or turn off logging for that application. An error (as opposed to a warning) is now reported in such cases only if there is no available output at all for ndb_mgmd event logging. (Bug #20960839)
- **Microsoft Windows:** MySQL NDB Cluster did not compile correctly with Microsoft Visual Studio 2015, due to a change from previous versions in the VS implementation of the _vsnprintf() function. (Bug #80276, Bug #22670525)
- During node failure handling, the request structure used to drive the cleanup operation was not
 maintained correctly when the request was executed. This led to inconsistencies that were harmless
 during normal operation, but these could lead to assertion failures during node failure handling, with
 subsequent failure of additional nodes. (Bug #22643129)
- The previous fix for a lack of mutex protection for the internal TransporterFacade::deliver_signal() function was found to be incomplete in some cases. (Bug #22615274)

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

- When setup of the binary log as an atomic operation on one SQL node failed, this could trigger a state in other SQL nodes in which they appeared to detect the SQL node participating in schema change distribution, whereas it had not yet completed binary log setup. This could in turn cause a deadlock on the global metadata lock when the SQL node still retrying binary log setup needed this lock, while another mysqld had taken the lock for itself as part of a schema change operation. In such cases, the second SQL node waited for the first one to act on its schema distribution changes, which it was not yet able to do. (Bug #22494024)
- Duplicate key errors could occur when ndb_restore was run on a backup containing a unique index. This was due to the fact that, during restoration of data, the database can pass through one or more inconsistent states prior to completion, such an inconsistent state possibly having duplicate values for a column which has a unique index. (If the restoration of data is preceded by a run with --disable-indexes and followed by one with --rebuild-indexes, these errors are avoided.)

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

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

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

- Optimization of signal sending by buffering and sending them periodically, or when the buffer became full, could cause SUB_GCP_COMPLETE_ACK signals to be excessively delayed. Such signals are sent for each node and epoch, with a minimum interval of TimeBetweenEpochs; if they are not received in time, the SUMA buffers can overflow as a result. The overflow caused API nodes to be disconnected, leading to current transactions being aborted due to node failure. This condition made it difficult for long transactions (such as altering a very large table), to be completed. Now in such cases, the ACK signal is sent without being delayed. (Bug #18753341)
- When setting CPU spin time, the value was needlessly cast to a boolean internally, so that setting it
 to any nonzero value yielded an effective value of 1. This issue, as well as the fix for it, apply both to
 setting the SchedulerSpinTimer parameter and to setting spintime as part of a ThreadConfig
 parameter value. (Bug #80237, Bug #22647476)
- A logic error in an if statement in storage/ndb/src/kernel/blocks/dbacc/DbaccMain.cpp rendered useless a check for determining whether ZREAD_ERROR should be returned when comparing operations. This was detected when compiling with gcc using -Werror=logical-op. (Bug #80155, Bug #22601798)

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

- Suppressed a CMake warning that was caused by use of an incorrectly quoted variable name. (Bug #80066, Bug #22572632)
- When using CREATE INDEX to add an index on either of two NDB tables sharing circular foreign
 keys, the query succeeded but a temporary table was left on disk, breaking the foreign key
 constraints. This issue was also observed when attempting to create an index on a table in the
 middle of a chain of foreign keys—that is, a table having both parent and child keys, but on different
 tables. The problem did not occur when using ALTER TABLE to perform the same index creation

operation; and subsequent analysis revealed unintended differences in the way such operations were performed by CREATE INDEX.

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

 The PortNumber SCI, SHM, and TCP configuration parameters, which were deprecated in MySQL 5.1.3, have now been removed and are no longer accepted in configuration files.

This change does not affect the PortNumber management node configuration parameter, whose behavior remains unaltered. (Bug #77405, Bug #21280456)

Changes in MySQL NDB Cluster 7.5.0 (5.7.10-ndb-7.5.0) (2016-02-05, Development Milestone)

- · Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Important Change: Previously, the ndbinfo information database included lookup tables that used the MyISAM storage engine. This dependency on MyISAM has now been removed. (Bug #20075747, WL #7575)
- Important Change: Previously, the NDB scheduler always optimized for speed against throughput in a predetermined manner (this was hard coded); this balance can now be set using the SchedulerResponsiveness data node configuration parameter. This parameter accepts an integer in the range of 0-10 inclusive, with 5 as the default. Higher values provide better response times relative to throughput. Lower values provide increased throughput, but impose longer response times. (Bug #78531, Bug #21889312)
- Important Change: A number of MySQL NDB Cluster data node configuration parameters were deprecated in earlier versions of MySQL NDB Cluster, and have been removed with this release. These parameters include Id, NoOfDiskPagesToDiskDuringRestartTUP, NoOfDiskPagesToDiskDuringRestartACC, NoOfDiskPagesToDiskAfterRestartACC, NoOfDiskPagesToDiskAfterRestartTUP, ReservedSendBufferMemory, MaxNoOfIndexes, and Discless (use Diskless instead), as well as DiskCheckpointSpeed and DiskCheckpointSpeedInRestart. The archaic and unused ByteOrder computer configuration parameter has also been removed, as well as the unused MaxNoOfSavedEvents management node confugration parameter. These parameters are no longer supported; most of them already did not have (or no longer had) any effect. Trying to use any of these parameters in a MySQL NDB Cluster configuration file now results in an error.

For more information, see What is New in NDB Cluster 7.5. (Bug #77404, Bug #21280428)

- Important Change: The ndbinfo database can now provide default and current information about MySQL NDB Cluster node configuration parameters as a result of the following changes:
 - 1. The config_params table has been enhanced with additional columns providing information about each configuration parameter, including its type, default, and maximum and minimum values (where applicable).
 - 2. A new config_values table has been added. A row in this table shows the current value of a parameter on a given node.

You can obtain values of MySQL NDB Cluster configuration parameters by name using a join on these two tables such as the one shown here:

SELECT p.param_name AS Name,

```
v.node_id AS Node,
    p.param_type AS Type,
    p.param_default AS 'Default',
    v.config_value AS Current

FROM config_params p

JOIN config_values v

ON p.param_number = v.config_param

WHERE p. param_name IN ('NodeId', 'HostName','DataMemory', 'IndexMemory');
```

(Bug #71587, Bug #18183958, WL #8703)

• Important Change: The ExecuteOnComputer configuration parameter for management, data, and API nodes is now deprecated, and is subject to removal in a future MySQL NDB Cluster version. For all types of MySQL NDB Cluster nodes, you should now use the HostName parameter exclusively for identifying hosts in the cluster configuration file.

This information is also now displayed in the output of ndb_config --configinfo --xml. (Bug #53052, Bug #11760628)

- Deprecated MySQL NDB Cluster node configuration parameters are now indicated as such by
 ndb_config --configinfo --xml. For each parameter currently deprecated, the corresponding
 cparam/> tag in the XML output now includes the attribute deprecated="true". (Bug
 #21127135)
- Added the --ndb-cluster-connection-pool-nodeids option for mysqld, which can be used to specify a list of nodes by node ID for connection pooling. The number of node IDs in the list must equal the value set for --ndb-cluster-connection-pool. (Bug #19521789)
- Added the PROMPT command in the ndb_mgm client. This command has the syntax PROMPT string, which sets the client's prompt to string. Issuing the command without an argument causes the prompt to be reset to the default (ndb_mgm>). See Commands in the NDB Cluster Management Client, for more information. (Bug #18421338)
- When the --database option has not been specified for ndb_show_tables, and no tables are found in the TEST_DB database, an appropriate warning message is now issued. (Bug #50633, Bug #11758430)
- The NDB storage engine now uses the improved records-per-key interface for index statistics introduced for the optimizer in MySQL 5.7. Some improvements due to this change are listed here:
 - The optimizer can now choose better execution plans for queries on NDB tables in many cases where a less optimal join index or table join order would previously have been chosen.
 - EXPLAIN now provides more accurate row estimates than previously.
 - Improved cardinality estimates can be obtained from SHOW INDEX.

(WL #8165)

Bugs Fixed

- Incompatible Change; NDB Cluster APIs: The pollEvents2() method now returns -1, indicating an error, whenever a negative value is used for the time argument. (Bug #20762291)
- Important Change; NDB Cluster APIs: Ndb::pollEvents() is now compatible with the TE_EMPTY, TE_INCONSISTENT, and TE_OUT_OF_MEMORY event types introduced in MySQL NDB Cluster 7.4.3. For detailed information about this change, see the description of this method in the MySQL NDB Cluster API Developer Guide. (Bug #20646496)
- Important Change; NDB Cluster APIs: Added the method

 Ndb::isExpectingHigherQueuedEpochs() to the NDB API to detect when additional, newer
 event epochs were detected by pollEvents2().

The behavior of Ndb::pollEvents() has also been modified such that it now returns NDB_FAILURE_GCI (equal to ~(Uint64) 0) when a cluster failure has been detected. (Bug #18753887)

• Important Change; NDB Cluster APIs: To release the memory used for dropped event operations, the event API formerly depended on pollEvents() and nextEvent() to consume all events possibly referring to the dropped events. This dependency between dropEventOperation() and the first two methods required the entire event buffer to be read before attempting to release event operation memory (that is, until successive calls to pollEvents() and nextEvent() returned no more events).

A related cleanup issue arose following the reset of the event buffer (when all event operations had previously been dropped), and the event buffer was truncated by the first <code>createEventOperation()</code> call subsequent to the reset.

To fix these problems, the event buffer is now cleared when the last event operation is dropped, rather than waiting for a subsequent create operation which might or might not occur. Memory taken up by dropped event operations is also now released when the event queue has been cleared, which removes the hidden requirement for consuming all events to free up memory. In addition, event operation memory is now released as soon as all events referring to the operation have been consumed, rather than waiting for the entire event buffer to be consumed. (Bug #78145, Bug #21661297)

- Important Change; NDB Cluster APIs: The MGM API error-handling functions ndb_mgm_get_latest_error(), ndb_mgm_get_latest_error_msg(), and ndb_mgm_get_latest_error_desc() each failed when used with a NULL handle. You should note that, although these functions are now null-safe, values returned in this case are arbitrary and not meaningful. (Bug #78130, Bug #21651706)
- Important Change; NDB Cluster APIs: The following NDB API methods were not actually implemented and have been removed from the sources:
 - Datafile methods: getNode(), setNode(), and getFileNo()
 - Undofile methods: getNode(), setNode(), and getFileNo()
 - Table methods: getObjectType() and setObjectType()
- **Important Change:** The options controlling behavior of NDB programs with regard to the number and timing of successive attempts to connect to a management server have changed as listed here:
 - The minimum value for the --connect-retry-delay option common to all NDB programs has been changed from 0 to 1; this means that all NDB programs now wait at least 1 second between successive connection attempts, and it is no longer possible to set a waiting time equal to 0.
 - The semantics for the --connect-retries option have changed slightly, such that the value of this option now sets the number of times an NDB program tries to connect to a management server. Setting this option to 0 now causes the program to attempt the connection indefinitely, until it either succeeds or is terminated by other means (such as kill).
 - In addition, the default for the --connect-retries option for the ndb_mgm client has been changed from 3 to 12, so that the minimum, maximum, and default values for this option when used with ndb_mgm are now exactly the same as for all other NDB programs.

The ndb_mgm --try-reconnect option, although deprecated in MySQL NDB Cluster 7.4, continues to be supported as a synonym for ndb_mgm --connect-retries to provide backwards compatibility. The default value for --try-reconnect has also been changed from

3 to 12, respectively, so that this option continues to behave in the exactly in the same way as --connect-retries.

(Bug #22116937)

- Important Change: In previous versions of MySQL NDB Cluster, other DDL operations could not be part of ALTER ONLINE TABLE ... RENAME (This was disallowed by the fix for BUG#16021021.) MySQL NDB Cluster 7.5 makes the following changes:
 - Support for the ONLINE and OFFLINE keywords, which was deprecated in MySQL NDB Cluster 7.3, is now removed, and use of these now causes a syntax error; the NDB storage engine now accepts *only* ALGORITHM = DEFAULT, ALGORITHM = COPY, and ALGORITHM = INPLACE to specify whether the ALTER operation is copying or in-place, just as in the standard MySQL Server.
 - NDB now allows ALTER TABLE ... ALGORITHM=COPYING RENAME.

(Bug #20804269, Bug #76543, Bug #20479917, Bug #75797)

References: See also: Bug #16021021.

- NDB Disk Data: A unique index on a column of an NDB table is implemented with an associated internal ordered index, used for scanning. While dropping an index, this ordered index was dropped first, followed by the drop of the unique index itself. This meant that, when the drop was rejected due to (for example) a constraint violation, the statement was rejected but the associated ordered index remained deleted, so that any subsequent operation using a scan on this table failed. We fix this problem by causing the unique index to be removed first, before removing the ordered index; removal of the related ordered index is no longer performed when removal of a unique index fails. (Bug #78306, Bug #21777589)
- NDB Cluster APIs: The binary log injector did not work correctly with TE_INCONSISTENT event type handling by Ndb::nextEvent(). (Bug #22135541)

References: See also: Bug #20646496.

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

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

- NDB Cluster APIs: Creation and destruction of Ndb_cluster_connection objects by multiple
 threads could make use of the same application lock, which in some cases led to failures in the
 global dictionary cache. To alleviate this problem, the creation and destruction of several internal
 NDB API objects have been serialized. (Bug #20636124)
- NDB Cluster APIs: When an Ndb object created prior to a failure of the cluster was reused, the event queue of this object could still contain data node events originating from before the failure. These events could reference "old" epochs (from before the failure occurred), which in turn could violate the assumption made by the nextEvent() method that epoch numbers always increase. This issue is addressed by explicitly clearing the event queue in such cases. (Bug #18411034)

References: See also: Bug #20888668.

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

Introduction of transporter polling also revealed a problem with missing mutex protection in the ndbcluster_binlog handler, which has been added as part of this fix. (Bug #79311, Bug #20957068, Bug #22224571, WL #8627)

- NDB Cluster APIs: After the initial restart of a node following a cluster failure, the cluster failure event added as part of the restart process was deleted when an event that existed prior to the restart was later deleted. This meant that, in such cases, an Event API client had no way of knowing that failure handling was needed. In addition, the GCI used for the final cleanup of deleted event operations, performed by pollEvents() and nextEvent() when these methods have consumed all available events, was lost. (Bug #78143, Bug #21660947)
- A serious regression was inadvertently introduced in MySQL NDB Cluster 7.4.8 whereby local checkpoints and thus restarts often took much longer than expected. This occurred due to the fact that the setting for MaxDiskWriteSpeedOwnRestart was ignored during restarts and the value of MaxDiskWriteSpeedOtherNodeRestart, which is much lower by default than the default for MaxDiskWriteSpeedOwnRestart, was used instead. This issue affected restart times and performance only and did not have any impact on normal operations. (Bug #22582233)
- The epoch for the latest restorable checkpoint provided in the cluster log as part of its reporting for EventBufferStatus events (see NDB Cluster: Messages in the Cluster Log) was not well defined and thus unreliable; depending on various factors, the reported epoch could be the one currently being consumed, the one most recently consumed, or the next one queued for consumption.

This fix ensures that the latest restorable global checkpoint is always regarded as the one that was most recently completely consumed by the user, and thus that it was the latest restorable global checkpoint that existed at the time the report was generated. (Bug #22378288)

• Added the --ndb-allow-copying-alter-table option for mysqld. Setting this option (or the equivalent system variable ndb_allow_copying_alter_table) to OFF keeps ALTER TABLE statements from performing copying operations. The default value is ON. (Bug #22187649)

References: See also: Bug #17400320.

- Attempting to create an NDB table having greater than the maximum supported combined width for all BIT columns (4096) caused data node failure when these columns were defined with COLUMN_FORMAT DYNAMIC. (Bug #21889267)
- Creating a table with the maxmimum supported number of columns (512) all using COLUMN_FORMAT DYNAMIC led to data node failures. (Bug #21863798)
- In a MySQL NDB Cluster with multiple LDM instances, all instances wrote to the node log, even inactive instances on other nodes. During restarts, this caused the log to be filled with messages from other nodes, such as the messages shown here:

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

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

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

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

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

The fix for this bug also fixes similar timeout issues for a number of other signals that are sent the DBDICT kernel block as part of DDL operations, including ALTER TAB REQ.

CREATE_INDX_REQ, DROP_FK_REQ, DROP_INDX_REQ, INDEX_STAT_REQ, DROP_FILE_REQ, CREATE_FILEGROUP_REQ, DROP_FILEGROUP_REQ, CREATE_EVENT, WAIT_GCP_REQ, DROP_TAB_REQ, and LIST_TABLES_REQ, as well as several internal functions used in handling NDB schema operations. (Bug #21277472)

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

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

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

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

References: See also: Bug #20481140.

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

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

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

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

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

 Memory allocated when obtaining a list of tables or databases was not freed afterward. (Bug #20234681, Bug #74510)

References: See also: Bug #18592390, Bug #72322.

• Added the BackupDiskWriteSpeedPct data node parameter. Setting this parameter causes the data node to reserve a percentage of its maximum write speed (as determined by the value of MaxDiskWriteSpeed) for use in local checkpoints while performing a backup.

BackupDiskWriteSpeedPct is interpreted as a percentage which can be set between 0 and 90 inclusive, with a default value of 50. (Bug #20204854)

References: See also: Bug #21372136.

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

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

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

- Using ndb_mgm STOP -f to force a node shutdown even when it triggered a complete shutdown of
 the cluster, it was possible to lose data when a sufficient number of nodes were shut down, triggering
 a cluster shutodwn, and the timing was such that SUMA handovers had been made to nodes already
 in the process of shutting down. (Bug #17772138)
- When using a sufficiently large value for TransactionDeadlockDetectionTimeout
 and the default value for sort_buffer_size, executing SELECT * FROM
 ndbinfo.cluster_operations ORDER BY transid with multiple concurrent conflicting or
 deadlocked transactions, each transaction having several pending operations, caused the SQL node
 where the query was run to fail. (Bug #16731538, Bug #67596)
- The ndbinfo.config_params table is now read-only. (Bug #11762750, Bug #55383)
- NDB failed during a node restart due to the status of the current local checkpoint being set but not as active, even though it could have other states under such conditions. (Bug #78780, Bug #21973758)
- ndbmtd checked for signals being sent only after a full cycle in run_job_buffers, which is
 performed for all job buffer inputs. Now this is done as part of run_job_buffers itself, which
 avoids executing for extended periods of time without sending to other nodes or flushing signals to
 other threads. (Bug #78530, Bug #21889088)
- When attempting to enable index statistics, creation of the required system tables, events and event subscriptions often fails when multiple mysqld processes using index statistics are started concurrently in conjunction with starting, restarting, or stopping the cluster, or with node failure handling. This is normally recoverable, since the affected mysqld process or processes can (and do) retry these operations shortly thereafter. For this reason, such failures are no longer logged as warnings, but merely as informational events. (Bug #77760, Bug #21462846)
- It was possible to end up with a lock on the send buffer mutex when send buffers became a limiting resource, due either to insufficient send buffer resource configuration, problems with slow or failing communications such that all send buffers became exhausted, or slow receivers failing to consume what was sent. In this situation worker threads failed to allocate send buffer memory for signals, and attempted to force a send in order to free up space, while at the same time the send thread was busy trying to send to the same node or nodes. All of these threads competed for taking the send buffer mutex, which resulted in the lock already described, reported by the watchdog as Stuck in Send. This fix is made in two parts, listed here:
 - 1. The send thread no longer holds the global send thread mutex while getting the send buffer mutex; it now releases the global mutex prior to locking the send buffer mutex. This keeps worker threads from getting stuck in send in such cases.
 - 2. Locking of the send buffer mutex done by the send threads now uses a try-lock. If the try-lock fails, the node to make the send to is reinserted at the end of the list of send nodes in order to be retried later. This removes the Stuck in Send condition for the send threads.

(Bug #77081, Bug #21109605)

Index

Symbols

- --bootstrap, 63, 127
- --connect-retries, 72, 136
- --connect-retry-delay, 72, 136
- --database, 72, 136
- --diff-default, 42, 108
- --disable-indexes, 18, 88
- --extra-partition-info, 52, 117
- --host, 63, 127
- --initial, 33, 49, 101, 114
- --initialize, 52, 63, 117, 127
- --ndb-default-column-format, 52, 69, 117, 133
- --ndb-log-fail-terminate, 25, 94
- --ndbcluster, 52, 117
- --nostart, 38, 105
- --port, 49, 114
- --print-sql-log, 52, 117
- --query-all, 42, 108
- --rebuild-indexes, 26, 95
- --restore-epoch, 30, 98
- --try-reconnect, 72, 136
- -a, 42, 108
- -flifetime-dse, 42, 108
- -Werror=logical-op, 69, 133
- .ctl files, 28, 49, 96, 114
- 32-bit, 69, 133
- _vsnprintf(), 69, 133
- ~Ndb, 63, 127
- ~Ndb(), 52, 117
- ~Ndb_cluster_connection(), 52, 117

Α

aborted transactions, 38, 105

AbortOption, 25, 94

ACC scan, 63, 127

ADD DATAFILE, 31, 99

add nodes, 52, 117

ALGORITHM, 72, 136

ALGORITHM=INPLACE, 52, 117

ALGORITHM=INPLACE REORGANIZE PARTITION, 52, 117

ALTER, 26, 95

ALTER TABLE, 47, 52, 63, 72, 112, 117, 127, 136

ALTER TABLE ... ADD UNIQUE KEY, 42, 108

ANALYZE TABLE, 33, 63, 72, 101, 127, 136

ANTLR, 25, 94

AnyValue, 39

API nodes, 47, 69, 112, 133

ApiConnectRecord::TF_LATE_COMMIT, 52, 117

ApiVerbose, 63, 127

apt, 42, 108

arbitration, 63, 127

arrays, 15, 85

assert, 6

AUTHENTICATION_PAM_LOG, 7

Auto-Installer, 49, 114

autoincrement, 31, 99 AUTO_INCREMENT, 39, 106 averages, 6

backup, 20, 23, 26, 30, 42, 52, 63, 72, 90, 92, 95, 108, 117, 127, 136 BackupDataBufferSize, 69, 133 BackupDiskWriteSpeedPct, 72, 136 backups, 47, 112 BatchByteSize, 9, 81 batched_key_access, 16, 86 binary log, 42 binlog, 52, 117 binlog injector, 49, 72, 114, 136 binlog injector thread, 52, 117 binlog thread, 63, 127 BitmaskImpl::setRange(), 37, 104 BLOB, 19, 23, 30, 31, 33, 35, 52, 89, 92, 99, 101, 102, 117 blob table, 52, 117 blobs, 22, 91 bulk updates, 42, 108

C API, 69 cascading_scans_count, 52, 117 case sensitivity, 23, 92 CFG_DB_NO_TRIGGERS, 52, 117 changes NDB Cluster, 79 character sets, 28, 96 charsets, 49, 114 close(), 31, 99 cluster failure and recovery, 72, 136 cluster log, 69, 72, 133, 136 ClusterJDatastoreException, 52, 117 ClusterJPA, 42, 108 ClusterTransactionImpl, 38, 105 cluster locks, 61, 125 CMake3, 30, 98 CM_REGREF, 18, 88 COLUMN_FORMAT, 30, 52, 69, 98, 117, 133 COLUMN_FORMAT DYNAMIC, 72, 136 COMMENT, 63, 127 compatibility, 72, 136 compiling, 11, 30, 37, 41, 42, 47, 49, 63, 69, 98, 104, 107, 108, 114, 127, computeXorChecksum(), 15, 85 concurrency, 42, 108 concurrent operations, 31, 99 concurrent transactions, 72, 136 concurrent trigger operations, 35, 102 condition pushdown, 21, 33, 90, 101 ConfigInfo.cpp, 69, 133 configuration, 42 configuration parameters, 72, 136 configurator, 63, 127 config_nodes table, 42, 108

config_params, 72, 136 config_values, 72, 136 connection pooling, 72, 136 connection records, 52, 117 connections, 8, 52, 80, 117 CONTINUEB signal, 16, 86 copy fragment replicas, 52, 117 CopyFrag, 11, 83 copying operations, 72, 136 correlation IDs, 31, 99 cp1252, 49, 114 CPU binding of the receive threads, 41, 107 CPU locking, 41, 107 CPU usage, 63, 127 cpubind_exclusive, 63, 127 cpuset_exclusive, 63, 127 cpustat, 31, 61, 99, 125 CREATE INDEX, 69, 133 CREATE LOGFILE GROUP, 63, 127 CREATE NODEGROUP, 19, 52, 89, 117 CREATE TABLE, 37, 42, 52, 104, 117 createEvent(), 72, 136 c_exec, 52, 117

D

data node, 39, 106 data node failure, 42, 108 data node failures, 47, 112 data node restarts, 63, 127 data node shutdown, 37, 104 data nodes, 9, 11, 14, 38, 42, 72, 81, 83, 85, 105, 108, 136 Datafile, 72, 136 Date, 25, 94 DBACC, 42, 61, 63, 69, 108, 125, 127, 133 Dbacc::ACCKEY_error, 63, 127 Dbacc::startNext(), 35, 102 DBDICT, 9, 16, 52, 63, 72, 81, 86, 117, 127, 136 DBDIH, 21, 47, 52, 69, 90, 112, 117, 133 DBLQH, 42, 52, 108, 117 Dblgh::sendKeyinfo20(), 39, 106 DBSPJ, 9, 31, 42, 52, 63, 81, 99, 108, 117, 127 Dbspj::execSIGNAL_DROPPED_REP(), 52, 117 Dbspj::execTRANSID_AI(), 52, 117 DBTC, 8, 30, 38, 41, 42, 52, 63, 69, 80, 98, 105, 107, 108, 117, 127, 133 Dbtc::execSIGNAL DROPPED REP(), 42, 108 DBTUP, 13, 35, 42, 52, 63, 84, 102, 108, 117, 127 DbtupVarAlloc, 63, 127 DBTUX, 41, 107 DBUTIL, 52, 117 DDL statements, 72, 136 deadlocks, 72, 136 debugging, 63, 127 DefaultOperationRedoProblemAction, 72, 136 definitions, 49, 114 DELETE, 19, 42, 89, 108 deprecation, 72, 136 deprecations and removals, 69, 72, 133, 136

Developer Studio, 42, 108 Dictionary, 72, 136 dictionary cache, 42, 108 DictTrace, 63, 127 DIGETNODESREF, 47, 112 DIH_SCAN_GET_NODES_CONF, 69, 133 DIH SCAN_GET_NODES_REF, 69, 133 DIH_SCAN_GET_NODES_REQ, 69, 133 DIH_SCAN_TAB_COMPLETE_REP, 69, 133 DIH_SCAN_TAB_REQ, 69, 133 disconnection, 33, 101 DISK_RECORDS, 9, 81 DIVERIFYREQ, 16, 86 dojo, 22, 23, 91, 92 do_poll(), 63, 127 drop events, 72 DROP INDEX, 5 drop index, 72, 136 DROP NODEGROUP, 11, 83 DROP TABLE, 33, 42, 63, 101, 108, 127 dropEvent(), 72, 136 dropEventOperation, 72, 136 DROP_TAB_REQ, 52, 117 DROP_TRIG_IMPL_REQ, 35, 102 DUMP, 52, 117 DUMP 11001, 16, 86 DUMP 7027, 41, 107 DUMP 9988, 26, 95 DUMP 9989, 26, 95 **DUMP codes, 41, 107** duplicate keys, 69, 133 duplicate weedout, 39, 106 DYNAMIC, 47, 112

Ε

EMPTY LCP, 52, 117 epochs, 30, 98 error 1419, 72, 136 error 240, 42, 108 error 746, 72, 136 Error 899, 49, 114 error handling, 19, 41, 42, 72, 89, 107, 108, 136 error log, 19, 89 error logs, 63, 127 error messages, 28, 96 error reporting, 52, 63, 117, 127 errors, 13, 30, 42, 47, 52, 63, 84, 98, 108, 112, 117, 127 ER_NO_REFERENCED_ROW_2, 35, 102 event buffer, 20, 63, 72, 90, 127, 136 event logging, 69, 133 event queue, 72, 136 event reports, 72, 136 Event::TableEvent, 72, 136 EventBufferStatus, 72, 136 EventBufferStatus2, 69, 133 EventBytesRecvdCount, 15, 85 EventData, 63, 127 examples, 52, 117

exceptional event types, 72, 136 executeFullyReplicatedTrigger(), 52, 117 ExecuteOnComputer, 72, 136 EXECUTE_DIRECT, 69, 133 execute_signals(), 42, 108 exit, 49, 114 EXPLAIN, 47, 72, 112, 136

F

failure handling, 72 files, 16, 86 FILES, 39, 106 FIXED, 30, 98 FLUSH_AI, 42, 108 forced shutdown, 63, 127 foreign keys, 9, 26, 35, 37, 42, 47, 49, 52, 63, 69, 81, 95, 102, 104, 108, 112, 114, 117, 127, 133 FOR_RA_BY_LDM, 52, 117 FOR_RP_BY_LDM, 52, 117 FOR_RP_BY_NODE, 52, 117 FragmentCountType, 52, 117 FRAGMENT_COUNT_TYPE, 63, 127 FSAPPENDREF, 52, 117 FTS, 5 FULLY REPLICATED, 49, 52, 63, 114, 117, 127

G

gcc, 11, 42, 108 GCC 6, 49, 114 GCI, 23, 52, 92, 117 GCI boundary, 37, 104 GCP, 26, 95 GCP stall, 11, 83 GENERATED, 61, 125 getColumn(), 42, 108 getConnectionPoolSessionCounts(), 63, 127 getGCIEventOperations(), 9, 81 GETTABINFOREQ, 63, 127 GET TABLEID REQ, 18, 88 glibc, 39, 106 global checkpoints, 9, 21, 72, 81, 90, 136 global schema lock, 28, 96 global schema locks, 63, 127 GOUP BY, 42, 108 grandchild, 49, 114 GROUP AFFINITY, 63, 127 GSN_TRANSID_AI, 14, 85

Н

hash maps, 52, 117
HashMap, 49, 114
ha_ndbcluster::copy_fk_for_offline_alter(), 37, 104
ha_ndbcluster::exec_bulk_update(), 42, 108
ha_ndbcluster::get_metadata(), 63, 127
heartbeat failure handling, 42, 108
HostName, 72, 136
host_info, 63, 127

ı

IF NOT EXISTS, 37, 104 Important Change, 15, 16, 30, 33, 42, 52, 61, 63, 69, 72, 85, 86, 98, 101, 108, 117, 125, 127, 133, 136 IN, 38, 105 IN(), 8, 80 in-place operations, 72, 136 Incompatible Change, 26, 63, 72, 95, 127, 136 index, 61, 125 index invalidation, 42, 108 index length, 33, 101 index statistics, 9, 11, 13, 28, 72, 81, 83, 84, 96, 136 IndexMemory, 61, 125 INFORMATION SCHEMA, 39, 106 INFORMATION_SCHEMA.TABLES, 33, 101 initial restart, 72, 136 InitialNoOfOpenFiles, 31, 99 injector_mutex, 49, 114 InnoDB, 6, 8 INPLACE, 52, 61, 117, 125 install_mcc, 63, 127 invalid configuration, 42, 108 invalidateLcpInfoAfterSr(), 72, 136 InvalidAttrInfo, 63, 127

J

Java 11 deprecation warnings, 25, 94 Java versions, 25, 94 job buffer, 41, 42, 107, 108 job buffer full, 35, 102 joins, 8, 80 JOIN_TAB, 39, 106 JSON, 63, 127

L

LATE COMMIT. 52, 117 LCP, 26, 33, 41, 52, 69, 72, 95, 101, 107, 117, 133, 136 LCP pause, 33, 101 LCPs, 72, 136 LCP FRAG REP, 47, 52, 112, 117 LDM, 26, 42, 63, 95, 108, 127 leaks, 52, 117 LGMAN, 41, 107 libclass-methodmaker-perl, 42, 108 libmysqlclient, 52, 117 libndbclient, 52, 117 libndbclient-devel, 35, 102 limitations (removal), 42, 108 lock contention, 31, 99 locking, 49, 52, 61, 114, 117, 125 locks, 26, 95 locks_per_fragment, 61, 125 lock_ndb_objects(), 52, 117 log files, 42, 108 logging, 8, 16, 25, 26, 41, 63, 72, 80, 86, 94, 95, 107, 127, 136 long signals, 30, 98 LongMessageBuffer, 47, 49, 112, 114

LONGVARBINARY, 38, 105 lookups, 42, 108 lookup_resume, 52, 117 LooseScan, 39, 106 lost connection, 63, 127 LQHKEYREF, 52, 117 LQHKEYREQ, 30, 52, 98, 117

M

macOS, 63, 127 malloc(), 52, 117 master takeover, 52, 117 materialized semijoin, 39, 106 MaxBufferedEpochs, 31, 69, 99, 133 MaxDiskWriteSpeedOtherNodeRestart, 72, 136 MaxDiskWriteSpeedOwnRestart, 72, 136 MaxNoOfExecutionThreads, 33, 42, 101, 108 MaxNoOfOpenFiles, 31, 99 MAX NDB NODE GROUPS, 52, 117 MAX NULL BITS, 72, 136 MAX_ROWS, 28, 52, 96, 117 MD5() encryption function, 7 memory allocation, 63, 127 memory usage, 23, 72, 92, 136 MemoryUsage, 52, 117 metadata, 37, 63, 104, 127 metadata locaks, 63, 127 metadata lock, 47, 112 metadata operations, 69, 133 MgmtSrvr::restartNodes(), 49, 114 Microsoft Windows, 7, 9, 49, 63, 69, 80, 81, 114, 127, 133 mmap(), 63, 127 MM GROWN, 52, 117 MM SHRINK, 52, 117 monitoring, 63, 127 mt-scheduler, 72, 136 mt.cpp, 42, 108 mt thr config.cpp, 61, 125 mt_thr_config.cpp::do_bind(), 42, 108 multiple LDMs, 72, 136 multiple mysqlds, 72, 136 multiple-statement transactions, 72, 136 multithreaded, 72, 136 MyISAM, 63, 72, 127, 136 MySQL Enterprise Monitor, 16, 86 MySQL NDB ClusterJ, 5, 9, 11, 25, 28, 33, 35, 37, 38, 41, 42, 52, 63, 81, 83, 94, 96, 101, 102, 104, 105, 107, 108, 117, 127 mysgl.ndb apply status, 72 mysqld, 20, 31, 33, 42, 47, 52, 61, 63, 72, 90, 99, 101, 108, 112, 117, 125, 127, 136 mysqld --initialize, 61, 125 mysqld shutdown, 63, 127 mysqldump, 5, 6 mysql_cluster_move_privileges(), 52, 117 mysql_cluster_privileges_are_distributed(), 52, 117 mysql_options(), 69 mysql_upgrade, 52, 63, 117 m_abort(), 52, 117

m_buffer, 42, 108
m_buffered_size, 42, 108
m_deferred, 52, 117
m_failure_detected, 72, 136
m_latestGCl, 72, 136
m_max_batch_size_bytes, 42, 108
m_max_warning_level, 52, 117
m_sending, 42, 108
m_sending_size, 42, 108

N

NDB Client Programs, 6, 6, 21, 22, 25, 80, 90, 91, 94 NDB Cluster, 5, 6, 6, 7, 8, 9, 11, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 25, 26, 28, 30, 31, 33, 35, 37, 38, 39, 41, 42, 47, 49, 52, 61, 63, 69, 72, 80, 80, 80, 81, 83, 84, 85, 85, 86, 88, 89, 90, 90, 91, 92, 94, 95, 96, 98, 99, 101, 102, 104, 105, 106, 107, 108, 112, 114, 117, 125, 127, 133, 136 NDB Cluster APIs, 9, 14, 16, 19, 22, 31, 38, 42, 47, 52, 61, 63, 72, 81, 85, 86, 89, 91, 99, 105, 108, 112, 117, 125, 127, 136 NDB Disk Data, 31, 33, 47, 63, 72, 99, 101, 112, 127, 136 NDB distributed triggers, 63, 127 Ndb object, 63, 127 NDB Replication, 21, 30, 33, 39, 42, 63, 72, 127 NDB schema operations, 72, 136 NDB Util, 61, 125 NDB\$BLOB. 33 ndb-allow-copying-alter-table, 72, 136 ndb-batch-size, 15, 85 ndb-common, 19, 89 ndb-update-minimal, 42 Ndb::doDisconnect(), 52, 117 Ndb::getNextEventOpInEpoch3(), 39 Ndb::isExpectingHigherQueuedEpochs(), 72, 136 Ndb::nextEvent(), 72, 136 Ndb::pollEvents(), 72, 136 Ndb::pollEvents2(), 72, 136 Ndb::setEventBufferQueueEmptyEpoch(), 52, 63, 117, 127 ndbcluster_binlog_wait(), 52, 117 ndbcluster_print_error(), 31, 99 ndbd, 15, 16, 49, 63, 72, 85, 86, 114, 127, 136 NdbDictionary, 69, 133 NdbDuration::microSec(), 52, 117 NdbEventBuffer, 15, 85 NDBFS, 39, 106 NdbfsDumpRequests, 31, 99 NdbIndexOperation, 63, 127 NdbIndexScanOperation::setBound(), 38, 105 ndbinfo, 16, 52, 61, 72, 86, 117, 125, 136 ndbinfo Information Database, 37, 42, 49, 104, 108, 114 ndbinfo tables, 63, 127 ndbinfo.cluster_locks, 19, 89 ndbinfo.cluster_operations, 72, 136 ndbinfo.config_params, 72, 136 ndbinfo.tc_time_track_stats, 63, 127 ndbinfo.threadstat, 52, 117 NDBJTie, 42, 108 ndbmemcache, 21, 22, 42, 90, 91, 108 ndbmtd, 42, 47, 49, 52, 63, 72, 108, 112, 114, 117, 127, 136

```
NdbObjectIdMap, 47, 112
NdbOperation::OperationType, 61, 125
NdbReceiver, 38, 105
NdbReceiverBuffer, 25, 94
ndbrequire(), 63, 127
NdbScanFilter, 33, 101
NdbScanOperation::lockCurrentTuple(), 61, 125
NDBT, 30, 98
NdbTable, 42, 108
NdbThread SetThreadPrio, 14, 85
NdbTransaction, 31, 52, 99, 117
NdbTransaction::setSchemaObjectOwnerChecks(), 52, 117
ndb_binlog_index, 63, 127
ndb binlog setup(), 69, 133
ndb_blob_tool, 25, 94
ndb_clear_apply_status, 72
Ndb cluster connection, 52, 72, 117, 136
Ndb cluster connection::get system name(), 42, 108
Ndb_cluster_connection::set_data_node_neighbour(), 52, 117
Ndb_cluster_connection::set_service_uri(), 42, 108
ndb config. 33, 42, 63, 72, 101, 108, 127, 136
ndb data node neighbour, 52, 63, 117, 127
ndb_desc, 20, 52, 61, 90, 117, 125
ndb_end(), 52, 117
ndb_eventbuffer_max_alloc, 42, 108
Ndb_free_list_t, 63, 127
ndb_fully_replicated, 63, 127
ndb index stat option, 72, 136
ndb init(), 63, 127
NDB_LE_EventBufferStatus, 19, 89
ndb_logevent_get_next(), 52, 117
Ndb logevent type, 19, 89
NDB_MAX_TUPLE_SIZE, 72, 136
ndb_mgm, 52, 72, 117, 136
ndb_mgmd, 9, 11, 16, 22, 52, 63, 69, 81, 83, 86, 91, 117, 127, 133
ndb_mgm_get_latest_error(), 72, 136
ndb_mgm_get_latest_error_desc(), 72, 136
ndb_mgm_get_latest_error_msg(), 72, 136
NDB_MGM_NODE_TYPE_UNKNOWN, 18, 88
ndb optimized node selection, 52, 117
ndb_print_backup_file, 47, 112
ndb_print_file, 63, 127
ndb read backup, 63, 127
ndb_redo_log_reader, 6, 80
ndb_report_thresh_binlog_epoch_slip, 42, 52, 108, 117
Ndb rep tab key, 19, 89
ndb restore, 6, 9, 18, 19, 21, 22, 25, 26, 28, 30, 31, 33, 37, 38,
49, 52, 63, 69, 81, 88, 89, 90, 91, 94, 95, 96, 98, 99, 101, 104,
105, 114, 117, 127, 133
ndb setup.pv. 21, 22, 90, 91
ndb_show_tables, 25, 42, 49, 72, 94, 108, 114, 136
Ndb system name, 42, 108
NDB TABLE, 41, 52, 61, 63, 107, 117, 125, 127
Ndb UnlockCPU(), 39, 106
Ndb_util_thread, 52, 117
ndb_waiter, 9, 25, 81, 94
nextEvent(), 72, 136
nice, 52, 117
```

node copy phase, 52, 117 node failure, 72, 136 node failure handling, 15, 21, 41, 42, 69, 85, 90, 107, 108, 133 node failures, 42, 108 node ID allocation, 28, 96 node IDs, 15, 85 node restart, 41, 52, 72, 107, 117, 136 node restarts, 23, 49, 52, 92, 114, 117 node shutdown, 11, 83 node start, 63, 127 Node.js, 22, 91 nodeFailure error, 63, 127 NodeGroup, 22, 91 Nodelnfo, 63, 127 NoOfFragmentLogParts, 42, 108 NOT_STARTED, 49, 114 **NOWAIT, 8, 80** NO_OF_BUCKETS, 18 NULL, 33, 38, 47, 72, 101, 105, 112, 136 null, 52, 117

0

object creation, 72, 136 object destruction, 72, 136 OFFLINE, 72, 136 offline mode, 6 OM_WRITE_BUFFER, 39, 106 ON DELETE CASCADE, 31, 99 ON UPDATE CASCADE, 49, 114 ONLINE, 52, 72, 117, 136 online operations, 42, 108 OpenSSL, 8, 80 open_table(), 37, 104 OPTIMIZE TABLE, 6 optimizer, 72, 136 ORDER BY, 39, 106 ordered index, 63, 127 ordered indexes, 52, 117 OS X, 69, 133 overflow, 52, 117 O_DIRECT, 22, 26, 52, 91, 95, 117 O SYNC, 39, 106

P

Packaging, 19, 22, 33, 35, 42, 47, 49, 52, 89, 91, 101, 102, 108, 112, 114, 117
packaging, 42
page length entries, 52, 117
parallel schema operations, 49, 114
parseTableInfo(), 63, 127
partial restarts, 26, 95
PartionCount, 20, 90
partition ID, 52, 117
PartitionBalance, 52, 117
Partitioning, 47, 112
partitioning, 52, 117
partitions, 63, 127

PARTITION_BALANCE, 37, 52, 104, 117 Performance, 15, 63, 85, 127 performance, 69, 133 PID, 49, 114 pollEvents(), 49, 72, 114, 136 pollEvents2(), 72, 136 polling, 52, 117 poll_owner, 47, 112 PortNumber, 69, 133 PREPARE_SEIZE_ERROR, 63, 127 PRIMARY KEY, 42, 108 processes, 9, 37, 81, 104 processes table, 42, 108 PROMPT, 72, 136 proxies_priv, 52, 117 pushdown joins, 42, 108 pycrypto, 49, 114 python-crypt, 49, 114 python-paramiko, 42, 108 P_TAIL_PROBLEM, 72, 136

Q

QEP_TAB, 39, 106 QMGR, 41, 107 query cache, 47, 112 query memory, 31, 99

R

race, 38, 105 race condition, 31, 99 rand(), 49, 114 range checks, 16, 86 range scans, 16, 86 read backup replica, 52, 63, 117, 127 read-only, 72 READ_BACKUP, 13, 52, 61, 63, 84, 117, 125, 127 READ_COMMITTED, 52, 117 read cost(), 16, 86 read length, 42, 108 realtime, 63, 127 realtime break, 9, 81 receive thread, 15, 52, 85, 117 receive thread activation threshold, 41, 107 receive threads, 72, 136 reconnection, 42, 108 redo, 72, 136 redo log, 11, 21, 31, 83, 90, 99 redo log file rotation, 42, 108 redo log part metadata, 39, 106 RedoOverCommitCounter, 26, 95 RedoOverCommitLimit, 26, 95 refactoring, 49, 114 REGCONF, 8, 80 **REGREQ, 8, 80** released objects, 63, 127 RENAME, 72, 136 REORGANIZE PARTITION, 11, 38, 42, 52, 83, 105, 108, 117 REORGANIZE PARTITIONS, 52, 117 replicas, 52, 117 Replication, 41 reporting, 63, 127 request distribution, 42, 108 **RESET REPLICA, 21** RESET SLAVE, 21, 72 resource usage, 52, 63, 117, 127 **RESTART, 33, 101** restarts, 28, 37, 39, 47, 52, 61, 63, 72, 96, 104, 106, 112, 117, 125, 127, 136 restore, 52, 72, 117, 136 result buffers, 9, 81 row ID, 33, 101 ROW_FORMAT, 52, 69, 117, 133 RPM, 49, 52, 114, 117 run_job_buffers, 72, 136 SafeCounter, 31, 99 scanIndex(), 38, 105 **SCANREQ, 9, 81** scans, 47, 112 SCAN_FRAGCONF, 61, 125 SCAN_FRAGREQ, 61, 125 SCAN NEXTREQ, 61, 125 SCAN TABREQ, 52, 117 SchedulerResponsiveness, 72, 136 schema distribution, 63, 127 schema distribution coordinator, 47, 112 schema events, 72 schema operations, 47, 112 schema ops, 63, 127 SCI, 69, 133 scratch buffer, 9, 81 SELECT, 26, 33, 95, 101 semijoin, 39, 106 send buffer, 38, 63, 72, 105, 127, 136 send buffers, 38, 47, 105, 112 send thread usage, 63, 127 send threads, 72, 136 sending signals, 72, 136 send_buffer::m_node_total_send_buffer_size, 42, 108 server_locks, 61, 125 service, 49, 114 SessionFactory, 42, 108 SET LOGLEVELORD, 16, 86 shared global memory, 63, 127 SHM, 69, 133 SHOW CREATE TABLE, 47, 52, 112, 117 SHOW INDEX, 72, 136 SHUTDOWN, 42, 108 shutdown, 61, 125 signal data, 18, 88 signals, 11, 14, 38, 49, 83, 85, 105, 114 SignalSender, 26, 95 singal IDs, 9, 81

slave SQL thread, 42

slave_parallel_workers, 42 SNAPSHOTSTART, 30 Solaris, 42, 47, 63, 108, 127 sort_buffer_size, 72, 136 SPARC, 49, 114 SparseBitmask::getBitNo(), 42, 108 spintime, 69, 133 SPJ, 16, 39, 42, 86, 106, 108 SQL, 49, 114 SQL node, 69, 133 SQL nodes, 19, 69, 89, 133 start, stop, restart, 72, 136 statistics, 15, 20, 85, 90 STOP -f, 72, 136 STOP BACKUP, 52, 117 stop GCI, 37, 104 StopOnError, 49, 114 Stuck in Send, 72, 136 SUB_GCP_COMPLETE_ACK, 47, 69, 112, 133 SUB_GCP_COMPLETE_REP, 52, 72, 117, 136 SUB STOP REQ, 35, 102 SUMA, 11, 13, 18, 35, 39, 69, 72, 83, 84, 88, 102, 106, 133, 136 symbol conflicts, 52, 117 SYSTAB_0, 31, 99 system restart, 42, 52, 108, 117 system tables, 16, 86 sysvar_audit_log_strategy, 5

Т

Table, 72, 136 table reorganization, 16, 28, 86, 96 Table::getColumn(), 22, 91 tableno, 39, 106 tables, 49, 114 Table_Map, 39 TAB SCANREF, 13, 84 TCKEYREQ, 30, 52, 98, 117 TCP, 69, 133 TC_COMMIT_ACK, 47, 112 TEXT, 31, 35, 52, 99, 102, 117 TE EMPTY, 63, 127 Thd_ndb::m_connect_count, 63, 127 thread contention, 72, 136 ThreadConfig, 33, 61, 63, 69, 101, 125, 127, 133 thread_prio, 63, 127 thrman.cpp, 52, 117 throughput, 52, 117 TimeBetweenEpochs, 69, 133 timeout, 47, 112 timeouts, 49, 52, 69, 72, 114, 117, 133, 136 **TIMESTAMP, 52, 117 TINYBLOB, 30, 98** TOTAL_BUCKETS_INIT, 72, 136 trace files, 52, 63, 117, 127 trace logs, 9, 81 traces, 49, 114 transaction ID, 52, 117 TransactionDeadlockDetectionTimeout, 72, 136

transactions, 15, 26, 85, 95
TRANSID_AI, 42, 108
TransporterFacade, 63, 127
TransporterRegistry::prepareSendTemplate(), 38, 105
TRANS_AI, 42, 108
triggers (NDB), 35, 102
troubleshooting, 63, 127
TRUNCATE, 39, 106
tuple corruption, 35, 102
type conversion, 30, 98

U

undo files, 33, 101
undo log, 63, 127
Undofile, 72, 136
UNIQUE, 52, 117
UNIQUE INDEX, 52, 117
unique index, 63, 72, 127, 136
unique indexes, 52, 63, 117, 127
unique keys, 69, 133
unit tests, 42, 108
unlock_ndb_objects(), 52, 117
unplanned shutdown, 49, 114
unqualified option, 42, 108
UPDATE CASCADE, 47, 112
upgrades, 23, 52, 63, 92, 117, 127

V

verifyVarSpace(), 63, 127 versioning, 52, 117 VIRTUAL, 30 virtual columns, 52, 117 Visual Studio, 7, 80 VS 2015, 69, 133

W

wait locks, 35, 102 warnings, 8, 80 WITH_UNIT_TESTS, 41, 107 work threads, 72, 136

X

XML, 72, 136