Project SnappyData™ - Community Edition

Release Notes

Software Release 1.1.1
August 2019

TIBCO®

Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND

CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

ANY SOFTWARE ITEM IDENTIFIED AS THIRD PARTY LIBRARY IS AVAILABLE UNDER

SEPARATE SOFTWARE LICENSE TERMS AND IS NOT PART OF A TIBCO PRODUCT. AS SUCH, THESE SOFTWARE ITEMS ARE NOT COVERED BY THE TERMS OF YOUR AGREEMENT WITH TIBCO, INCLUDING ANY TERMS CONCERNING SUPPORT, MAINTENANCE, WARRANTIES, AND INDEMNITIES. DOWNLOAD AND USE OF THESE ITEMS IS SOLELY AT YOUR OWN DISCRETION AND SUBJECT TO THE LICENSE TERMS APPLICABLE TO THEM. BY PROCEEDING TO DOWNLOAD, INSTALL OR USE ANY OF THESE ITEMS, YOU ACKNOWLEDGE THE FOREGOING DISTINCTIONS BETWEEN THESE ITEMS AND TIBCO PRODUCTS.

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, the TIBCO logo, the TIBCO O logo, TIBCO ComputeDB, SnappyData, and Snappy are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. Please see the readme.txt file for the availability of this software version on a specific operating system platform.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS.

Contents

Overview

SnappyData™ is a memory-optimized database based on Apache Spark. It delivers very high throughput, low latency, and high concurrency for unified analytic workloads that may combine streaming, interactive analytics, and artificial intelligence in a single, easy to manage distributed cluster.

SnappyData offers a fully functional core OSS distribution, which is the **Community Edition**, that is Apache 2.0 licensed. The **Enterprise Edition** of the product, which is sold by TIBCO Software under the name **TIBCO ComputeDB™**, includes everything that is offered in the OSS version along with additional capabilities that are closed source and only available as part of a licensed subscription.

New Features

SnappyData 1.1.1 includes the following new features:

Full Support for External Hive Metastore

- Now, you can bring up a cluster pointing to an external Hive2 compatible metastore and immediately have access to all the tables that are defined in the catalog from SnappySession. There is no need to define the external tables explicitly. This can be enabled dynamically in a SnappySession and works in both Embedded and Smart connector modes.
- With this change, access to data sets in any compatible Hadoop distribution (For example EMR, Cloudera CDH, Horton HDP, Apache Hadoop) is much simpler than before.
- You can import external data into SnappyData in-memory tables, run joins across in-memory, and externally managed tables etc.
- You can also create tables registered in the external Hive catalog directly, like creating new Hive tables or views, to the extent Spark supports it.
- Finally, users can consume external libraries exposed as Hive UDFs.

Support for Approximate Query Processing (AQP) from TIBCO Spotfire®

- If samples are available on large in-memory or external tables, queries from Spotfire® can automatically use them. This permits true interactive latencies for large scale aggregations in Spotfire® without having to materialize the full data set in-memory.

Stability and Performance Improvements

SnappyData 1.1.1 includes the following stability and performance improvements:

- There is 15% performance improvement in the TPC-H benchmark at scale-factor 100 (with 100 GB data), over the 1.1.0 release. The results were obtained by setting the flag **snappydata.sql.useOptimizedHashAggregateForSingleKey=true** to avoid a known issue in Q17 (SNAP-3150). It is recommended to turn on this flag to avoid any such issues that affect the cluster stability until the known issue is fixed.

- When ingesting large data sets, Spark relies heavily on buffering large blocks of data in JVM memory. This stresses the system and the potentially resulting causes an OOM error. SnappyData now automatically detects such an overload and dynamically adjusts the **task count** used for loading. This prominently reduces the potential for such problems. [SNAP-3111]

- Aggregations are now even faster and do not stress the JVM Garbage collector. Earlier SnappyData used hash-based in-memory structures for aggregations. The in-memory structure was using JVM memory and subject to **stop the world** GC conditions. Now, the algorithms used are vectorized and rely mostly on off-heap memory. [SNAP-3053]

Enhancements

- SnappyData Quickstart examples can be imported into IDE as a separate project, modified and re-built with ease. [SNAP-3026]
- Added support for PIVOT in SQL. (snappydata PR #1395)
- Added support for **RESTRICT** and **CASCADE** constructs in **ALTER TABLE ... DROP COLUMN**. [SNAP-2482]
- Added support for interval expressions. Now expressions such as **timestamp + interval hour(time) hours** can be used. (snappydata PR #1360)
- Significant sections of the documentation rewritten or added. For example, you can see the complete list of SQL functions supported in the product without having to go through Spark manuals.
- Improvements to SnappyData monitoring console:
    - Any redundancy configuration for tables is now visible. [SNAP-1338][SNAP-2779]
    - Cluster start time and uptime are now visible.[SNAP-2604][SNAP-3087]
- It is now simpler to provision config changes consistently across many nodes in the network using a command [SNAP-3135]
- Improved support for Hive Thrift Server2.
- Support for connection failover for ODBC clients is added. ODBC clients can point to the cluster's locator, which then redirects the client to the appropriate server based on load.
- Added support for **PUT INTO ... VALUES** for column tables. [SNAP-2879]
- Spark compatibility fixes. (snappydata PR #1310)

Resolved Issues

SnappyData 1.1.1 includes the following resolved issues:

- Stabilized the JDBC Prepared Statement APIs with several important fixes. [SNAP-3082][SNAP-3065][SNAP-2254][SNAP-2644][SNAP-2765][SNAP-3007]
- Better handling of disk-full conditions during transactions. [SNAP-3061]
- The write operations on column tables are serialized. [SNAP-2828]
- Resolved catalog inconsistency issues for sample tables. [SNAP-3110]
- Several fixes for **[UN]DEPLOY JAR|PACKAGE**. [SNAP-2772][SNAP-3134][SNAP-2355]
  - Fixed issue in re-deploy (**UNDEPLOY** followed by **DEPLOY**) without needing to restart the cluster.
  - Failure to load a deployed jar (for example, due to unavailability of jar path) upon restart does not fail the cluster restart.
  - Unique alias name and names with hyphens and dots supported.
  - Better cleanup of jar artifacts after they are undeployed.
- Several fixes in UDF functionality. [SNAP-2636][SNAP-3069]
  - Fixed issue in re-creating a UDF (**DROP FUNCTION** followed by **CREATE FUNCTION** with the same name) without needing to restart the cluster.
  - Better cleanup of UDF artifacts after they are dropped.
- Added a fix for adding the primary key to a row table using **ALTER TABLE** DDL. [SNAP-3108]
- New property **allow-explicit-commit** is added to avoid transaction-related errors when working with SQL tools. [SNAP-3088]
- Fixed failure in the concurrent submission of Spark apps. [SNAP-3103]
- **REMOVE_METASTORE_ENTRY** system procedure is added to allow users to clean up a problematic table entry from Hive meta-store to enable cluster restart. [SNAP-3055]
- The startup script now ensures that leads are always started after at least one server is in running state. (snappydata PR #1413)
- **CREATE INDEX** on the table fails when database/schema for index and that of table do not match. This issue is now fixed. [SNAP-2885]
- Handled **CatalogStaleException** with retries in snappy sink. [SNAP-3023]
- Smart connector insert/update/delete plans do not refresh meta-data. This issue is now fixed. [SNAP-2228]

- Converting DataFrame to dataset fails in an Embedded job. This issue is now fixed. [SNAP-3028]

- Running query using **snappysession.sql()** in Smart Connector mode fails due to the stale catalog. This issue is now fixed. [SNAP-3024]

- Improved **CAST** behavior to throw an exception in query and other places. [SNAP-2052]

- Disabled Janino debugging by default to avoid temp Janino files from being generated. [SNAP-3010]

- **DELETE** followed by **INSERT** on the same key within a transaction fails when the unique index is present on the table. This issue is now fixed. [SNAP-2620]

- Masked sensitive information in **DESCRIBE EXTENDED** ... command. [SNAP-2653]

Known Issues

SnappyData 1.1.1 includes the following known issues:

Key	Item	Description	Workaround
SNAP-1422	Catalog in Smart connector inconsistent with servers.	Catalog in Smart connector inconsistent with servers when a table is queried from spark-shell (or from an application that uses Smart connector mode) the table metadata is cached on the Smart connector side. If this table is dropped from SnappyData Embedded cluster (by using snappy-shell, or JDBC application, or a Snappy job), the metadata on the Smart connector side stays cached even though catalog has changed (table is dropped). In such cases, the user may see unexpected errors such as `org.apache.spark.sql.AnalysisException : Table `SNAPPYTABLE` already exists` in the Smart connector app side, for example, for DataFrameWriter.saveAsTable() API if the same table name that was dropped is used in saveAsTable().	• User may either create a new SnappySession in such scenarios. **Or** • Invalidate the cache on the Smart Connector mode. For example, by calling `snappy.sessionCatalog .invalidateAll()`.
SNAP-1153	Creating a temporary table with the same name as an existing table in any schema should not be allowed.	When creating a temporary table, the SnappyData catalog is not referred, which means, a temporary table with the same name as that of an existing SnappyData table can be created. Two tables with the same name lead to ambiguity during query execution and can either cause the query to fail or return wrong results.	Ensure that you create temporary tables with a unique name.
SNAP-2910	DataFrame API behavior in Spark, Snappy.	Saving a Dataset using Spark's JDBC provider with SnappyData JDBC driver into SnappyData row/column tables fails.	Use row or column provider in the Embedded or Smart connector. For Spark versions not supported by Smart connector, use the SnappyData JDBC Extension Connector.
SNAP-3148	Unicode escape character `'\u'` does not work for `"insert into table values()"` syntax.	Escape character `'\u'` is used to indicate that code following `'\u'` is for a unicode character.	As a workaround, instead of `"insert into table values ('\u...')"` syntax, use `"insert into table select '\u...' '"` syntax. User can also directly insert the unicode char instead of using escape sequence. For example: `create table region (val string, description string) using column` Following insert query will not insert unicode char corresponding to code `'\u7ca5'`. Instead it will insert a string

Key	Item	Description	Workaround

			value `'\u7ca5'`
			
			insert into region values (`'\u7ca5',`
			`'unicode2')`
			
			However, following insert statement will insert the appropriate unicode char
			
			insert into region select `'\u7ca5',`
			`'unicode2'`
			
			Following query that directly inserts a unicode char instead of using escape char also works.
			
			insert into region values (`'粵' ,`
			`'unicode')`

[SNAP-3146](#)	UDF execution from Smart Connector.	A UDF once executed from smart connector side continues to remain accessible from the same SnappySession on Smart connector side, even if it is deleted from embedded side.	Drop the UDF from Smart connector side or use a new SnappySession.
[SNAP-3150](#)	With default configuration for release 1.1.1, some queries such as TPC-H Q17 can cause servers to crash.	With the current product default configuration for release 1.1.1, some queries that perform an aggregate operation, such as group by/ sum/ avg etc., on a single key where the key column has high cardinality can cause one or more data servers and sometimes the lead to crash.	To avoid this issue, set the property `snappydata.sql.useOptimizedHashAggregateForSingleKey=true` in the Lead configuration file.