YugabyteDB

Integrating Spark SQL and Yugabyte Cloud with a Scala sample application

Summary

Yugabyte Cloud is a fully managed YugabyteDB-as-a-Service that allows you to run YugabyteDB clusters on Google Cloud Platform (GCP) and Amazon Web Services (AWS).

This demo covers building and deploying a sample Scala application to demonstrate integration between Spark SQL and Yugabyte cloud via YCQL API. Yugabyte Cloud Query Language (YCQL) is a semi-relational SQL API that is best fit for internet-scale OLTP and HTAP applications needing massive data ingestion and blazing-fast queries. It supports strongly consistent secondary indexes, a native JSON column type, and distributed transactions. It has its roots in the Cassandra Query Language (CQL).

Prerequisites

- Apache Spark 3.0.3 installed
- Yugabyte Cloud cluster registered
- Yugabyte Spark Cassandra connector: **3.0-yb-**8
- Scala version: 2.12.10 installed
- Sbt 1.5.5 installed
- JDK 1.8 installed

Yugabyte cluster setup in Yugabyte Cloud

Go to https://www.yugabyte.com/cloud/ and follow the instructions:

- Create a free cluster and config network access accordingly
- Download the certificate and connect to the database as follows

```
SSL_CERTFILE=~/spark3yb/root.crt ./bin/ycqlsh 96d0e0a4-954b-4b02-8fea-b0ab1b03689a.aws.ybdb.io 9042 -u admin --ssl
```

- Create a namespace and tables in Yugabyte cloud using YCQL API

```
admin@ycqlsh: Create namespace test;
admin@ycqlsh:Use test ;
CREATE TABLE employees_json_index
```

```
 (department_id INT,
  employee_id INT,
  dept_name TEXT,
  salary Double,
  phone jsonb,
  PRIMARY KEY(department_id, employee_id))
   with transactions = { 'enabled' : true };

INSERT INTO employees_json_index(department_id, employee_id, dept_name, salary, phone) VALUES (1, 1, 'Sales', 10000,
'{"code":"+1","phone":7462505400}');
INSERT INTO employees_json_index(department_id, employee_id, dept_name, salary, phone) VALUES (1, 2, 'Sales', 10000,
'{"code":"+1","phone":5122505400}');
INSERT INTO employees_json_index(department_id, employee_id, dept_name, salary, phone) VALUES (2, 3, 'IT', 20000,
'{"code":"+1","phone":5122505400}');
INSERT INTO employees_json_index(department_id, employee_id, dept_name, salary, phone) VALUES (2, 4, 'IT', 20000,
'{"code":"+1","phone":5122555400}');
INSERT INTO employees_json_index(department_id, employee_id, dept_name, salary, phone) VALUES (3, 5, 'HR',
30000,'{"code":"+1","phone":5172505400}');
```

Building a Scala application to integrate Spark and Yugabyte cloud

- Create project directory tree: Spark3yb

  ```
  mkdir ~/spark3yb | cd ~/spark3yb

  mkdir -p src/main/scala | mkdir project | mkdir target
  ```

- Create a text file named build.sbt and add following contents

  ```
  cd ~/spark3yb
  ```

  vi build.sbt

  ```
  name := "spark3yb"

  version := "0.1"

  scalaVersion := "2.12.10"

  scalacOptions := Seq("-unchecked", "-deprecation")

  val sparkVersion = "3.0.1"

  // maven repo at https://mvnrepository.com/artifact/com.yugabyte.spark/spark-cassandra-connector
  libraryDependencies += "com.yugabyte.spark" %% "spark-cassandra-connector-assembly" % "3.0-yb-8"

  // maven repo at https://mvnrepository.com/artifact/org.apache.spark/spark-core
  libraryDependencies += "org.apache.spark" %% "spark-core" % sparkVersion

  // maven repo at https://mvnrepository.com/artifact/org.apache.spark/spark-sql
  libraryDependencies += "org.apache.spark" %% "spark-sql" % sparkVersion
  ```

Also create a text file named assembly.sbt under project and add the following line

```
vi ~/spark3yb/project/assembly.sbt
 addSbtPlugin("com.eed3si9n" % "sbt-assembly" % "0.14.10")
```

- Create a sample Scala application

```
cd src/main/scala
```

vi spark3yb.scala

```
package com.yugabyte.sample.apps

import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.Row
import com.datastax.spark.connector._
import org.apache.spark.sql.cassandra.CassandraSQLRow
import org.apache.spark.sql.cassandra._
import com.datastax.spark.connector.cql.CassandraConnectorConf
import org.apache.spark.sql.functions._
import org.apache.spark.sql.expressions.Window

object spark3yb {

 def main(args:Array[String]): Unit = {

val host = "748fdee2-aabe-4d75-a698-a6514e0b19ff.aws.ybdb.io"
val keyspace = "test"
val table = "employees_json"
val user = "admin"
val password = "UMhJyvWzh5wb%JmJrJelHPyY"
val keyStore ="/Users/xxx/Documents/spark3yb/yb-keystore.jks"

// Setup the local spark master
val conf = new SparkConf()
   .setAppName("yb.spark-jsonb")
   .setMaster("local[1]")
   .set("spark.cassandra.connection.localDC", "us-east-2")
   .set("spark.cassandra.connection.host", "127.0.0.1")
   .set("spark.sql.catalog.ybcatalog", "com.datastax.spark.connector.datasource.CassandraCatalog")
   .set("spark.sql.extensions", "com.datastax.spark.connector.CassandraSparkExtensions")

val spark = SparkSession
    .builder()
    .config(conf)
    .config("spark.cassandra.connection.host", host)
    .config("spark.cassandra.connection.port", "9042")
    .config("spark.cassandra.connection.ssl.clientAuth.enabled", true)
    .config("spark.cassandra.auth.username", user)
```

```
    .config("spark.cassandra.auth.password", password)
    .config("spark.cassandra.connection.ssl.enabled", true)
    .config("spark.cassandra.connection.ssl.trustStore.type", "jks")
    .config("spark.cassandra.connection.ssl.trustStore.path", keyStore)
    .config("spark.cassandra.connection.ssl.trustStore.password", "ybcloud")
    .withExtensions(new CassandraSparkExtensions)
    .getOrCreate()

 //example with Spark.sql
runReadWriteSqlExample(spark)

spark.stop()
 }

 private def runReadWriteSqlExample(spark: SparkSession): Unit = {

//List namespace
spark.sql("SHOW NAMESPACES FROM ybcatalog").show

//Creating Data Frame by reading testing data from YB cloud database

val df_yb = spark.read.table("ybcatalog.test.employees_json")

df_yb.printSchema()
df_yb.count()
df_yb.show()

//another example: Loading an Dataset using a format helper and a option helper, equivalent to the previous example
val df_yb = spark
 .read
 .cassandraFormat("employees_json", "test")
 .options(ReadConf.SplitSizeInMBParam.option(32))
 .load()

//Performing ETL : Window function
//row_number()
val windowSpec  = Window.partitionBy("department_id").orderBy("salary")
//ranking: row_number() window function is used to give the sequential row number starting from 1 to the result of each
//window partition.
 df_yb.withColumn("row_number",row_number.over(windowSpec)).show()

//rank() window function is used to provide a rank to the result within a window partition. This function leaves gaps in rank
when there are ties.
df_yb.withColumn("rank",rank().over(windowSpec)).show()
println("Read/Write successful")

 //writing back  to YCQL: Persisting a Dataset to Database using Save command, following examples are equivalent
 df.write
      .cassandraFormat("personcopy", "test")
      .mode("overwrite")
      .save()
//To verify
```

```
val sqlDF = spark.sql("SELECT * FROM ybcatalog.test.employees_json_copy").show(false)


//Native support of Json:
val df = spark.sql("SELECT * FROM ybcatalog.test.employees_json WHERE get_json_object(phone, '$.phone') = 1200");
df.show

//Using JSONB Column Pruning
val query = "SELECT department_id, employee_id, get_json_object(phone, '$.code') as code FROM
ybcatalog.test.employees_json WHERE get_json_string(phone, '$.key(1)') = '1400' order by department_id limit 2";
val df_sel1=spark.sql(query)
Df_sel.explain

//Predicate pushed down
val query = "SELECT department_id, employee_id, get_json_object(phone, '$.key[1].m[2].b') as key FROM
ybcatalog.test.employees_json WHERE get_json_string(phone, '$.key[1].m[2].b') = '1400' order by department_id limit 2";

val df_sel2 = spark.sql(query)
df_sel2.show()
df_sel2.explain
println("Json processing successful")
 }
```

- Build and run it:

```
cd ~/spark3yb

#remove any prior build | Compile into Java byte code

sbt clean | sbt compile

# Create jar

sbt package

# run it

./bin/spark-submit --conf spark.cassandra.connection.host=127.0.0.1  --conf
spark.sql.extensions=com.datastax.spark.connector.CassandraSparkExtensions --packages
com.yugabyte.spark:spark-cassandra-connector_2.12:3.0-yb-8 --class
com.yugabyte.sample.apps.spark3yb ~/spark3yb/target/scala-2.12/spark3yb_2.12-0.1.jar
```

Source Code References

Example Spark applications are available in our Github repository at:

- Spark 3 tests and Spark 3 sample apps
- Specifically for JSONB processing there is: TestSpark3Jsonb.java.