Hadoop and Spark Developer - CCA 175

Problem Scenario 4

PLEASE READ THE INTRODUCTION TO THIS SERIES. CLICK ON HOME LINK AND READ THE INTRO BEFORE ATTEMPTING TO SOLVE THE PROBLEMS

Video walk-through of the solution to this problem can be found here ***[Click here]***

Click here for the video version of this series. This takes you to the youtube playlist of videos.

In this problem, we will focus on conversion between different file formats using spark or hive. This is a very import examination topic. I recommend that you master the data file conversion techniques and understand the limitations. You should have an alternate method of accomplishing a solution to the problem in case your primary method fails for any unknown reason. For example, if saving the result as a text file with snappy compression fails while using spark then you should be able to accomplish the same thing using Hive. In this blog\video I am going to walk you through some scenarios that cover alternative ways of dealing with same problem.

Problem 4:

1. Import orders table from mysql as text file to the destination **/user/cloudera/problem5/text.** Fields should be terminated by a tab character ("\t") character and lines should be terminated by new line character ("\n").
2. Import orders table from mysql into hdfs to the destination **/user/cloudera/problem5/avro**. File should be stored as avro file.
3. Import orders table from mysql into hdfs to folders **/user/cloudera/problem5/parquet**. File should be stored as parquet file.
4. Transform/Convert data-files at **/user/cloudera/problem5/avro** and store the converted file at the following locations and file formats
   - save the data to hdfs using snappy compression as parquet file at **/user/cloudera/problem5/parquet-snappy-compress**
   - save the data to hdfs using gzip compression as text file at **/user/cloudera/problem5/text-gzip-compress**
   - save the data to hdfs using no compression as sequence file at **/user/cloudera/problem5/sequence**
   - save the data to hdfs using snappy compression as text file at **/user/cloudera/problem5/text-snappy-compress**
5. Transform/Convert data-files at **/user/cloudera/problem5/parquet-snappy-compress** and store the converted file at the following locations and file formats
   - save the data to hdfs using no compression as parquet file at **/user/cloudera/problem5/parquet-no-compress**
   - save the data to hdfs using snappy compression as avro file at **/user/cloudera/problem5/avro-snappy**
6. Transform/Convert data-files at **/user/cloudera/problem5/avro-snappy** and store the converted file at the following locations and file formats
   - save the data to hdfs using no compression as json file at **/user/cloudera/problem5/json-no-compress**
   - save the data to hdfs using gzip compression as json file at **/user/cloudera/problem5/json-gzip**
7. Transform/Convert data-files at **/user/cloudera/problem5/json-gzip** and store the converted file at the following locations and file formats
   - save the data to as comma separated text using gzip compression at **/user/cloudera/problem5/csv-gzip**
8. Using spark access data at **/user/cloudera/problem5/sequence** and stored it back to hdfs using no compression as ORC file to HDFS to destination **/user/cloudera/problem5/orc**

Solution:

Try your best to solve the above scenario without going through the solution below. If you could then use the solution to compare your result. If you could not then I strongly recommend that you go through the concepts again (this time in more depth). Each step below provides a solution to the points mentioned in the Problem Scenario.

Step 1:

```
sqoop import --connect "jdbc:mysql://quickstart.cloudera:3306/retail_db" --password cloudera --username retail_dba --table orders --as-textfile
--fields-terminated-by '\t' --target-dir /user/cloudera/problem5/text -m 1
```

Step 2:

```
sqoop import --connect "jdbc:mysql://quickstart.cloudera:3306/retail_db" --password cloudera --username retail_dba --table orders --as-
avrodatafile--target-dir /user/cloudera/problem5/avro -m 1
```

Step 3:

```
sqoop import --connect "jdbc:mysql://quickstart.cloudera:3306/retail_db" --password cloudera --username retail_dba --table orders --as-
parquetfile --target-dir /user/cloudera/problem5/parquet -m 1
```

Step 4:

```
var dataFile = sqlContext.read.avro("/user/cloudera/problem5/avro");
sqlContext.setConf("spark.sql.parquet.compression.codec","snappy");
dataFile.repartition(1).write.parquet("/user/cloudera/problem5/parquet-snappy-compress");
dataFile.map(x=> x(0)+"\t"+x(1)+"\t"+x(2)+"\t"+x(3)).saveAsTextFile("/user/cloudera/problem5/text-gzip-
compress",classOf[org.apache.hadoop.io.compress.GzipCodec]);
dataFile.map(x=> (x(0).toString,x(0)+"\t"+x(1)+"\t"+x(2)+"\t"+x(3))).saveAsSequenceFile("/user/cloudera/problem5/sequence");

Below may fail in some cloudera VMS. If the spark command fails use the sqoop command to accomplish the problem. Remember you need
to get out to spark shell to run the sqoop command.

dataFile.map(x=> x(0)+"\t"+x(1)+"\t"+x(2)+"\t"+x(3)).saveAsTextFile("/user/cloudera/problem5/text-snappy-
compress",classOf[org.apache.hadoop.io.compress.SnappyCodec]);

sqoop import --table orders --connect "jdbc:mysql://quickstart.cloudera:3306/retail_db" --username retail_dba --password cloudera --as-textfile
-m1 --target-dir user/cloudera/problem5/text-snappy-compress --compress --compression-codec
org.apache.hadoop.io.compress.SnappyCodec
```

Step 5:

```
var parquetDataFile = sqlContext.read.parquet("/user/cloudera/problem5/parquet-snappy-compress")
sqlContext.setConf("spark.sql.parquet.compression.codec","uncompressed");
parquetDataFile.write.parquet("/user/cloudera/problem5/parquet-no-compress");
sqlContext.setConf("spark.sql.avro.compression.codec","snappy");

parquetDataFile.write.avro("/user/cloudera/problem5/avro-snappy");
```

Step 6:

```
var avroData = sqlContext.read.avro("/user/cloudera/problem5/avro-snappy");
avroData.toJSON.saveAsTextFile("/user/cloudera/problem5/json-no-compress");
avroData.toJSON.saveAsTextFile("/user/cloudera/problem5/json-gzip",classOf[org.apache.hadoop.io.GzipCodec]);
```

Step 7 :

```
var jsonData = sqlContext.read.json("/user/cloudera/problem5/json-gzip");
jsonData.map(x=>x(0)+","+x(1)+","+x(2)+","+x(3)).saveAsTextFile("/user/cloudera/problem5/csv-gzip",classOf[org.apache.hadoop.io.compress.GzipCodec])
```

Step 8:

```
//To read the sequence file you need to understand the sequence getter for the key and value class to //be used while loading the sequence file as a spark RDD.
//In a new terminal Get the Sequence file to local file system
hadoop fs -get /user/cloudera/problem5/sequence/part-00000
//read the first 300 characters to understand the two classes to be used.
cut -c-300 part-00000

//In spark shell do below
var seqData =
sc.sequenceFile("/user/cloudera/problem5/sequence/",classOf[org.apache.hadoop.io.Text],classOf[org.apache.hadoop.io.Text]);
seqData.map(x=>{var d = x._2.toString.split("\t"); (d(0),d(1),d(2),d(3))}).toDF().write.orc("/user/cloudera/problem5/orc");
```

Contribution from **Raphael L. Nascimento:**

You can use below method as well for setting the compression codec.
sqlContext.sql("SET spark.sql.parquet.compression.codec=snappy")

29 comments:

Raphael L. Nascimento May 19, 2017 at 9:40 AM

Hi Arun, firstly I'd like to thank you for your great work here, these problems are very helpful.

I have one question:

After set the compress for snappy using (sqlContext.setConf("spark.sql.parquet.compress.codec","snappy")).

And writing the content as parquet (orders.write.parquet("/user/cloudera/problem4/parquet-snappy-compress")).

It seems the content was stored with default compression (gz).

hdfs dfs -ls /user/cloudera/problem4/parquet-snappy-compress

part-r-00009-681f9ff9-1709-4513-bec7-01787ff85b09.gz.parquet

Did you face the same issue ?

I currently using cloudera-quickstart-vm-5.8.0-0, spark version 1.6.0.

Best Regards
Raphaeel Nascimento

Reply

Replies

Arun Kumar Pasuparthi May 19, 2017 at 10:01 AM

Hi Raphael. You are facing the issue because of mis spelled word. the key is spark.sql.parquet.compression.codec and not spark.sql.parquet.compress.codec. So changing from compress to compression should solve your problem. Also, i recommend that you remember these names and values of the keys as they come in very handy during the exam.

Thank you for the nice complement though.

Raphael L. Nascimento May 19, 2017 at 10:07 AM

I'm so sorry, thank you. There is so many packages to remember and configs, I think I got confused with the package name of the compression codecs, which use the word "compress".

Tks.

Arun Kumar Pasuparthi May 19, 2017 at 10:14 AM

No problem. BTW, why did you delete your comment. The alternate procedure you showed is also very valid. May be you activate your comment again so that the views can benefit from using both the ways. I am giving credit to you and posting your way as well in a few minutes. Check the blog in 10 minutes or so you see your procedure as well reflected in the solution section

Reply

Raphael L. Nascimento May 19, 2017 at 10:01 AM

This comment has been removed by the author.

Reply

Raphael L. Nascimento May 19, 2017 at 10:20 AM

Besides the fact I made the confusion with compress word, there is a alternative way to set the compression codec:

sqlContext.sql("SET spark.sql.parquet.compression.codec=snappy")

Regards
Raphael.

Reply

Raphael L. Nascimento May 19, 2017 at 10:46 AM

Another approach to save time in test to transform DF in RDD is (depends on the number of dataframe' columns):

Use the concat_ws sql function.

Pyhton:

from pyspark.sql.functions import *

orders.select(concat_ws("\t", *orders.columns)).map(lambda rec: (rec[0])).saveAsTextFile("path", "compress-codec")

orders.select(orders[0], concat_ws("\t", *orders.columns)).map(lambda rec: (int(rec[0]), rec[1])).saveAsSequenceFile("path", "compress-codec")

Regards
Raphael.

Reply

laxman dashree May 24, 2017 at 4:17 PM

Hi Arun,

Thanks for the blog.
How is an ORC file saved with compression ?

Is this valid ?
sqlContext.setConf("spark.sql.orc.compression.codec","SomeCodec")
SomeDF.write.orc("somePath")

It does write the file in ORC format but the data seems to be in uncompressed format without any compression codec extension.

Can you please clarify this ?

Regards,

Reply

Replies

Arun Kumar Pasuparthi May 25, 2017 at 9:09 AM

there is no configuration called spark.sql.orc.compression.codec that i am aware of

instead of below

sqlContext.setConf("spark.sql.orc.compression.codec","SomeCodec")

Can you try below instead and let me know?

sqlContext.setConf("spark.io.compression.codec","snappy")

laxman dashree May 25, 2017 at 9:36 PM

Thanks for the suggestion, but the compression wasn't applied i guess.
Please see the hdfs file size with and without snappy compression.

[cloudera@quickstart ~]$ hadoop fs -ls /user/cloudera/CCApractice/problem4/orc_snappy_arun
Found 2 items
-rw-r--r-- 1 cloudera cloudera 0 2017-05-25 21:32 /user/cloudera/CCApractice/problem4/orc_snappy_arun/_SUCCESS
-rw-r--r-- 1 cloudera cloudera 306065 2017-05-25 21:32 /user/cloudera/CCApractice/problem4/orc_snappy_arun/part-r-00000-3fe70373-dd2c-414f-80e6-5f2f9c300fc9.orc
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ hadoop fs -ls /user/cloudera/CCApractice/problem4/orc
Found 2 items
-rw-r--r-- 1 cloudera cloudera 0 2017-05-19 15:16 /user/cloudera/CCApractice/problem4/orc/_SUCCESS
-rw-r--r-- 1 cloudera cloudera 306065 2017-05-19 15:16 /user/cloudera/CCApractice/problem4/orc/part-r-00000-b86a666c-9ad2-4ca9-b82f-f917089a7660.orc
[cloudera@quickstart ~]$

vishvas patel July 20, 2017 at 11:27 AM

In ORC compression if we don't specify any compression codec than by default it will get SnappyCompression as per given below code of apache-spark (2.X) version
https://github.com/apache/spark/blob/master/sql/hive/src/main/scala/org/apache/spark/sql/hive/orc/OrcOptions.scala

Reply

vishvas patel June 8, 2017 at 12:10 PM

Can you explain me why in (K,V) pair, you cast as string instead of Integer ?
dataFile.map(x=> (x(0).toString,x(0)+"\t"+x(1)+"\t"+x(2)+"\t"+x(3))).saveAsSequenceFile("/user/cloudera/problem5/sequence");

Reply

pd June 9, 2017 at 12:39 AM

Hi Arun,

Thanks for the consolidated material, this is a real confidence booster.

I have a small doubt:
In step 6, I'm trying the following solution to write the compressed json file:
val avroDF = sqlContext.read("/user/cloudera/problem5/avro-snappy")
sqlContext.setConf("set spark.sql.json.compression.codec", "Gzip");
avroDF.write.json("/user/cloudera/problem5/json-compress")

It is not compressing the json file . Do you see any error in the above written commands ?

Reply

Replies

akhil November 10, 2017 at 6:27 PM

Try avroDF.toJSON.saveAsTextFile("/user/cloudera/problem5/json-gzip",classOf[org.apache.hadoop.io.compress.GzipCodec])

Worked for me!

Reply

Arun Kumar Pasuparthi June 9, 2017 at 4:45 AM

Please take a look at this link where I consolidated all file handling mechanisms. Jason compression is also noted here

http://arun-teaches-u-tech.blogspot.com/p/file-formats.html?m=1

Reply

chandru G June 23, 2017 at 9:14 PM

Hi Arun,

You blog is just amazing! I am finding it really useful to study and practice everything in one place.

Step8 of this problem in which we have to store the result in ORC.. I get this error...any help is appreciated.

scala> ordersSeq.map(r => { var a = r._2.toString.split("\t") ; (a(0),a(1),a(2),a(3))}).toDF().write.orc("/user/cloudera/problem4/orc")
java.lang.AssertionError: assertion failed: The ORC data source can only be used with HiveContext.
at scala.Predef$.assert(Predef.scala:179)

Are you planning to come up with any more videos/problem scenarios in this series?

Reply

Skanda July 3, 2017 at 6:48 AM

Hi Arun,

Any idea how we can convert a csv to a tab delimited format?

Reply

Replies

Arun Kumar Pasuparthi July 3, 2017 at 8:39 AM

Read as text, split every record with ',', and then map each records elements by concatenating with a tab. store the file as text

Reply

YUVARAJ July 8, 2017 at 4:41 PM

This comment has been removed by the author.

Reply

YUVARAJ July 8, 2017 at 4:43 PM

Hi Arun,
I am unable to save ORC files in compressed formats.
I have tried the following

sqlContext.setConf("orc.compress","SNAPPY")
avroData.write.orc("/user/cloudera/test/orc-snappy")

Can you please help me out with this ?

Reply

Sagar Bunny July 14, 2017 at 2:31 PM

As you have told that we can get result from any format and the result is important right! ---- can i do that all compressions from sqoop instead of spark as it is much easy from sqoop compare to spark

Reply

Sagar Bunny July 14, 2017 at 2:34 PM

This comment has been removed by the author.

Reply

Unknown July 20, 2017 at 11:01 AM

This comment has been removed by the author.

Reply

Pankaj Tiwary July 20, 2017 at 1:39 PM

I would like to thank Arun for his great effort. This is very useful to prepare CCA 175.
Today I cleared CCA 175 with 8 out of 9 questions.
This playlist gave me immense confidence to clear CCA 175 .
Thanks again Arun!!

Reply

Replies

Arun Kumar Pasuparthi July 21, 2017 at 5:31 AM

Congratulations

Reply

Mayur Bhagia August 10, 2017 at 2:57 AM

Hello Arun. Thanks very much for the content and efforts that you have put-in for benefit of aspirants like me. One query for problem scenario 4 - step 4 - item a - is it sqlContext.setConf("spark.sql.parquet.compression.codec","snappy"); or sqlContext.setConf("spark.sql.parquet.compress.codec","snappy"); As per blog it is compression.codec and as per video it is compress.codec and i tried both, the parquet file with snappy compression of size 270k gets generated when we use compress.codec, however when we use compression.codec, the file still remains of 476k. Please advise. Thanks.

Reply

Revathi K September 21, 2017 at 1:08 PM

Hi,

step 5: converting parquet to avro with snappy compression

I did the following,
spark-shell --packages com.databricks:spark-avro_2.10:2.0.1 --master yarn --conf spark.ui.port=12345
import com.databricks.spark.avro._
val parquetSnappy= sqlContext.read.parquet("/user/rkathiravan/fileformats/ex/ord_avro_parquet_snappy")
sqlContext.setConf("spark.sql.avro.compression.codec","snappy")
parquetSnappy.write.avro("/user/rkathiravan/fileformats/ex/ord_avro_parquet_snappy_to_avro_snappy1")

it runs successfully but

in that file location
[rkathiravan@gw01 ~]$ hadoop fs -ls /user/rkathiravan/fileformats/ex/ord_avro_parquet_snappy_to_avro_snappy1
Found 5 items
-rw-r--r-- 3 rkathiravan hdfs 0 2017-09-21 15:27 /user/rkathiravan/fileformats/ex/ord_avro_parquet_snappy_to_avro_snappy1/_SUCCESS
-rw-r--r-- 3 rkathiravan hdfs 195253 2017-09-21 15:27 /user/rkathiravan/fileformats/ex/ord_avro_parquet_snappy_to_avro_snappy1/part-r-00000-17e6c260-7622-41ba-b715-a3d3df2846e0.avro
-rw-r--r-- 3 rkathiravan hdfs 169247 2017-09-21 15:27 /user/rkathiravan/fileformats/ex/ord_avro_parquet_snappy_to_avro_snappy1/part-r-00001-17e6c260-7622-41ba-b715-a3d3df2846e0.avro
-rw-r--r-- 3 rkathiravan hdfs 193189 2017-09-21 15:27 /user/rkathiravan/fileformats/ex/ord_avro_parquet_snappy_to_avro_snappy1/part-r-00002-17e6c260-7622-41ba-b715-a3d3df2846e0.avro
-rw-r--r-- 3 rkathiravan hdfs 172071 2017-09-21 15:27 /user/rkathiravan/fileformats/ex/ord_avro_parquet_snappy_to_avro_snappy1/part-r-00003-17e6c260-7622-41ba-b715-a3d3df2846e0.avro

i dont see whether it is snappy compressed .
can anyone help me out.

Reply

Replies

Bala September 23, 2017 at 4:24 AM

Snappy Compressed AVRO file does not have a "snappy" in its file name. You can compare the file size of the uncompressed avro file and confirm if your compression has worked fine or it. You can also refer to Arun's video.

Reply

Unknown December 20, 2017 at 10:01 AM

Hi Arun, Is there any specific reason you used 1 mapper in all your sqoop imports

Reply

Home

Subscribe to: Posts (Atom)

If you have landed on this page then you are most likely aspiring to learn Hadoop ecosystem of technologies and tools. Why not make you...

(no title)
If you have landed on this page then you are most likely aspiring to learn Hadoop ecosystem of technologies and tools. Why not make you...