Apache Spark- AAT Notes

Python, Scala, pyspark scala shell and apps.

Arturo Alatriste Trujillo

Abstract: this document is a cookbook of easy quick recipes for using Apache Spark, Scala, python, shells, configurations...

Also, there is a section for some topics for databricks cloud servers.

Contents

Hadoop File System

How to copy file to Hadoop File System

```
$ hadoop fs -copyFromLocal README.txt /user/cloudera/README.txt
```

How to list files in Hadoop File System

```
$ hadoop fs -ls /user/cloudera
```

Spark Shell

How to load Spark Shell (Scala)

Find the spark-shell path, and type something like this

```
$ /usr/bin/spark-shell
Or
$ ./usr/bin/spark-shell
```

You will get something like

```
scala>
```

Configure log4j.properties

Set verbose level

Set the property rootCategory to any of this levels

- ALL
- TRACE
- DEBUG
- INFO
- WARN
- ERROR
- FATAL
- OFF

See example below

```
# Set everything to be logged to the console

#log4j.rootCategory=INFO, console
```

```
#log4j.rootCategory=DEBUG, console
log4j.rootCategory=WARN, console

log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.target=System.err
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{1}: %m%n

# Settings to quiet third party logs that are too verbose
log4j.logger.org.eclipse.jetty=WARN
log4j.logger.org.eclipse.jetty.util.component.AbstractLifeCycle=ERROR
log4j.logger.org.apache.spark.repl.SparkIMain$exprTyper=ERROR
log4j.logger.org.apache.spark.repl.SparkILoop$SparkILoopInterpreter=ERROR
```

Spark Applications - Run locally

How to run a python script

```
$ spark-submit wordcount.py shakespeare.txt
```

How to run a scala and java applications

Compile

```
$ mvn package
```

How to run a jar

```
$ spark-submit
  --class myFolder.WordCount
  Wordcount-1.0.jar shakespeare.txt
```

Spark Applications - Run in cluster

To run your spark application in the cluster, include the parameter master

Start spark-master service

```
$ sudo service spark-master start
```

Start spark-worker service

```
$ sudo service spark-worker start
```

Stop spark-master service

```
$ sudo service spark-master stop
```

Stop spark-worker service

```
$ sudo service spark-worker stop
```

How to run in cluster a python script

```
$ spark-submit
--master spark://localhost/7077
--name   'Count JPG images requests'
count_jpg.py /loudacre/weblogs/*
```

How to run in cluster a scala / java applications

```
$ spark-submit
--class  stub.CountJPGs
--master spark://localhost/7077
--name   'Count JPG images requests'
target/countjpgs-1.0.jar /loadacre/weblogs/*
```

Monitoring

Monitor master in internet browser

```
http://localhost:18080
```

Monitor worker in internet browser

```
http://localhost:18081
```

Monitor in YARN Resource Manager

```
http://localhost:4040
```

Monitor in HUE

```
http://localhost:8888
```

Monitor in Apache Spark UI

```
http://localhost:4040
```

Monitor stages in internet browser

```
http://localhost:4040
http://localhost:4040/stages
```

Import Libraries for SQL and Dataframes

```
// SQLContext entry point for working with structured data
val sqlContext = new org.apache.spark.sql.SQLContext( sc )

// this is used to implicitly convert an RDD to a DataFrame.
import sqlContext.impIicits._

// this is used to implicitly convert an RDD to a DataFrame.
import sqlContext.implicits._

// Import Spark SQL data types and Row.
import org.apache.spark.sql._

// Import Spark SQL functions.
import org.apache.spark.sql.functions
```

Load Data

How to convert array to parallel data

```
val data = Array(1, 2, 3, 4, 5)
val distData = sc.parallelize(data)
```

How to load a File from Hadoop File System

First copy your file to Hadoop File System, as shown above.

```
val txt_df = sc.textFile( “hdfs://quickstart.cloudera:8020
/user/cloudera/README.txt” )
```

How to load file from Databricks File System

```
val log_file_path = "dbfs:/databricks-datasets/cs100/lab2/data-
001/apache.access.log.PROJECT"
val base_df       = sqlContext.read.text( log_file_path )
```

How to load file from File System

```
val log_file_path = " file:/databricks/driver/books.txt"
val base_df       = sqlContext.read.text( log_file_path )
```

How to count the number of rows in an RDD

```
scala> txt_df.count()
```

Data Frames

Convert from dataFrame to RDD

```
df.rdd
```

Convert from RDD to dataFrame

```
import sqlContextimplicits._
val df_1 = rdd.toDF()

val df_2 = SparkSession.createDataFrame( rdd )
```

Print, show rows from a dataFrame

```
df.show( n = num_rows, truncate = False )


// do not truncate columns to fit the screen
df.show( truncate = false)
```

Display as table in notebook

```
display( myDataFrame )
```

Print Show schema

```
myDataFrame.printSchema()
```

Show name of columns

```
myDataFrame.columns
```

Rename columns

First use printSchema to see the names of the columns

```
myDataFrame.printSchema()

output:

root:
        _1:string( nullable = true )
        _2:string( nullable = true )
        _3:string( nullable = true )
```

Then use “as” to rename the column as we use to do in sql.

```
myDataFrame.selectExpr( “_1 as name”, “_2 as email”, “_3 as favoriteBook” )
```

next check using printSchema

```
myDataFrame.printSchema()

output:

root:
        name          :string( nullable = true )
        email         :string( nullable = true )
        favouriteBook:string( nullable = true )
```

SELECT (object syntax)

```
val df_1 = df.select( "name" )
val df_2 = df.select( $"name", $"age" + 1 )
val df_4 = df.select( $"name".alias( “NAME” ), $"age".alias( “AGE” ) )
```

SELECT (sql syntax)

```
// register table to make queries
df.createOrReplaceTempView( "df" )

// execute query
val df_1 = sqlContext.sql(
  """
  SELECT *
  FROM df
  WHERE
    city = ‘Bangkok’
  """ )

// the number of bad_rows should be cero
df_1.show()
```

Filter, when

```
// year-month-day
df.filter( " my_time_col = '1995-08-03' " ).show()
```

Load data from file

load from text file

```
val base_df = sqlContext.read.text( log_file_path )
```

load from text file – infer schema

```
spark.read.csv(
    "some_input_file.csv", header=True, mode="DROPMALFORMED", schema=schema
)
```

or

```
(spark.read
    .schema(schema)
    .option("header", "true")
    .option("mode", "DROPMALFORMED")
    .csv("some_input_file.csv"))
```

load from text file – specify schema - scala

```
f
```

load from text file – specify schema-python

```
from pyspark.sql.types import StructType, StructField
from pyspark.sql.types import DoubleType, IntegerType, StringType

schema = StructType([
    StructField("A", IntegerType()),
    StructField("B", DoubleType()),
    StructField("C", StringType())
])

(sqlContext
    .read
    .format("com.databricks.spark.csv")
    .schema(schema)
    .option("header", "true")
    .option("mode", "DROPMALFORMED")
    .load("some_input_file.csv"))
```

load table from Hive

```
// use a spark session variable, here is spark.
// then the sql method and write your query
val diamonds = spark.sql("SELECT * FROM diamonds")

// in databricks you can call directly the display command
display( diamonds )


display(diamonds.select("*"))
```

create new dataframe with select statement

```
val new_df = df.select( )
```

Save as csv to file system

```
df.write.format( "com.databricks.spark.csv" ).save( "file:/tmp1/results.csv"
)
```

Save as parquet to file system

```
df.write.format( "parquet"                  ).save( "file:/tmp1/
results.parquet" )
```

Save as csv to dbfs

```
df.write.format( "com.databricks.spark.csv" ).save( "/tmp1/results.csv"     )
```

Save as parquet to dbfs

```
df.write.format( "parquet"                  ).save( " /tmp1/ results.parquet"
)
```

Save dataframe using columns for partitions

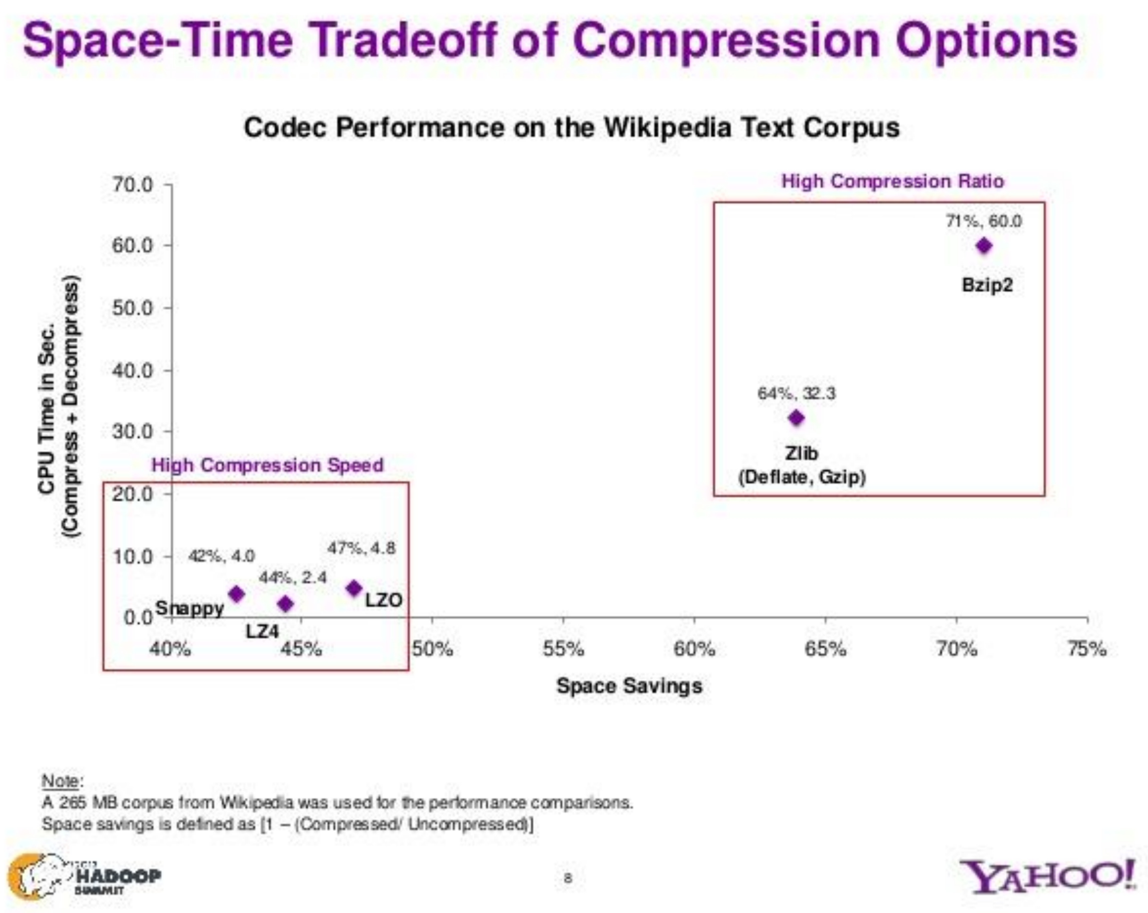
```
df
.write
.partitionBy( "end_year", "end_month" )
.parquet( "/tmp/sample_table" )
```

Set Compression Codec

```
# Accepted Compression Codec values: uncompressed, snappy, gzip, lzo

# set Compression Codec
sqlContext.setConf("spark.sql.parquet.compression.codec", "gzip")

# save data
df.write.format( "parquet" ).save( "/myFolder/data.parquet" )
```



Note:
A 265 MB corpus from Wikipedia was used for the performance comparisons.
Space savings is defined as [1 – (Compressed/ Uncompressed)]

filtering

```
// suppose a dataframe df, with columns name, age, address, tel, sex, etc...

// show all the girls
df.filter( “sex == ‘woman’ ).count()


// count the number of adults
df.filter( “age > 18” ).count()

// show people without address
df.filter( “address is null” ).show
```

Aggregation function on DataFrame

```
val status_to_count_df = logs_df
                       .groupBy( "status" )
                       .count()
                       .sort( "status" )
                       .cache()
```

```
Found 7 response codes

+------+------+

|status| count|

+------+------+

| 200  |940847|

| 302  | 16244|

| 304  | 79824|

| 403  | 58   |

| 404  | 6185 |

| 500  | 2    |

| 501  | 17   |

+------+------+
```

Aggregation function on DataFrame – Count number of rows

```
val content_size_stats =  logs_df.agg(
                            min( $"content_size" ),
                            avg( $"content_size" ),
                            max( $"content_size" ) )
                       .first()
```

Aggregation function on DataFrame – Sum values in column

```
//
```

Aggregation function on DataFrame – count number of null values in each column (object syntax)

```
import org.apache.spark.sql.functions.{sum,when}

split_df.agg(
   sum( when( $"host"         .isNull, 1 ).otherwise( 0 ) ).as( "host"
),
   sum( when( $"timestamp"    .isNull, 1 ).otherwise( 0 ) ).as( "timestamp"
),
   sum( when( $"path"         .isNull, 1 ).otherwise( 0 ) ).as( "path"
),
   sum( when( $"status"       .isNull, 1 ).otherwise( 0 ) ).as( "status"
),
   sum( when( $"content_size" .isNull, 1 ).otherwise( 0 ) ).as(
"content_size" )
 ).show()
```

Aggregation function on DataFrame – count number of null values in each column (SQL syntax)

```
val bad_rows_df3 = sqlContext.sql(
  """
  SELECT
    sum ( CASE WHEN host         is null THEN 1 ELSE 0 END ) as host,
    sum ( CASE WHEN timestamp    is null THEN 1 ELSE 0 END ) as timestamp,
    sum ( CASE WHEN path         is null THEN 1 ELSE 0 END ) as path,
    sum ( CASE WHEN status       is null THEN 1 ELSE 0 END ) as status,
    sum ( CASE WHEN content_size is null THEN 1 ELSE 0 END ) as content_size
  FROM split_df
  """ )

// the number of bad_rows should be cero
bad_rows_df3.show()
```

UDF

```
// create a Scala function (only if you want)
def upper_fun( s : String ) : String =
{
  println( "running upper_fun" )
  val r = s.toUpperCase
  return r
}

// create a scala expression
val upper = (s : String) => upper_fun( s )

// create udf
val upperUDF = udf(upper)

cleaned_df.withColumn( "HOST_Upper", upperUDF( $"host" ) ).show(  )
```

Column

Convert a column from string to category

```
import org.apache.spark.ml.feature.StringIndexer

val df = spark.createDataFrame(
  Seq((0, "a"), (1, "b"), (2, "c"), (3, "a"), (4, "a"), (5, "c"))
).toDF("id", "category")

val indexer = new StringIndexer()
  .setInputCol("category")
  .setOutputCol("categoryIndex")

val indexed = indexer.fit(df).transform(df)
indexed.show()
```

Dates

dayofmonth

```

val day_to_host_pair_df = logs_df
                          .select( $"host", dayofmonth( $"time" ).alias(
"day" )  )
```

Conversions

The string is in the yyyy-MM-dd format.

Convert String to Date – object syntax

```
/// import org.apache.spark.sql.functions.unix_timestamp

cleaned_month.select(
              $"host"   ,
              unix_timestamp( $"string_column",
"dd/MM/yyyy").cast("timestamp").alias( "new_column_timestamp"    ),
              $"path"         ,
              $"status"       ,
              $"content_size"
             )
```

Convert String to Date – sql syntax

```
val tmp1 = sqlContext.sql(
  """
  SELECT
    host,
    TO_DATE(CAST(UNIX_TIMESTAMP( string_column, 'dd/MM/yyyy') AS TIMESTAMP))
AS new_column_timestamp,
    path,
    status,
    content_size
  FROM cleaned_month
  """
)
```

Databricks

https://community.cloud.databricks.com

Download a file

```
%sh curl -O 'https://github.com/databricks/spark-
xml/raw/master/src/test/resources/books.xml'
```

Checkout where it saves it

```
%fs ls "file:/databricks/driver"
```

How to load file from databricks file system

```
val log_file_path = "dbfs:/databricks-datasets/cs100/lab2/data-
001/apache.access.log.PROJECT"
val base_df = sqlContext.read.text( log_file_path )
```

references

sql

http://stackoverflow.com/questions/39727742/how-to-filter-out-a-null-value-from-spark-dataframe

http://stackoverflow.com/questions/41765739/count-the-number-of-non-null-values-in-a-spark-dataframe
http://stackoverflow.com/questions/5487892/sql-server-case-when-or-then-else-end-the-or-is-not-supported
http://stackoverflow.com/questions/30783517/apache-spark-add-an-case-when-else-calculated-column-to-an-existing-d

udfs

https://docs.databricks.com/spark/latest/spark-sql/udf-scala.html

https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-sql-udfs.html

Dates

http://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.sql.functions$

http://stackoverflow.com/questions/40763796/convert-date-from-string-to-date-format-in-dataframes

http://stackoverflow.com/questions/36948012/how-to-change-the-column-type-from-string-to-date-in-dataframes

http://stackoverflow.com/questions/40844171/scala-convert-string-to-date-in-apache-spark

http://stackoverflow.com/questions/29844144/better-way-to-convert-a-string-field-into-timestamp-in-spark

http://stackoverflow.com/questions/34408183/spark-scala-dataframe-timestamp-conversion-sorting

File Storage

https://docs.databricks.com/user-guide/advanced/filestore.html

https://docs.databricks.com/user-guide/dbfs-databricks-file-system.html

Data Frames

https://stackoverflow.com/questions/29383578/how-to-convert-rdd-object-to-dataframe-in-spark

Getting Started (include predefined variables)

https://docs.databricks.com/user-guide/getting-started.html

Convert column from string to category

https://spark.apache.org/docs/latest/ml-features.html#stringindexer

Data Rows

https://stackoverflow.com/questions/35720330/getting-specific-field-from-chosen-row-in-pyspark-dataframe