Spark2 Workflow Scheduling for hue Integrated Oozie Workflow

Keywords: Big Data Spark GreenPlum SQL sudo

I. Environmental preparation

CDH5.15.0，spark2.3.0，hue3.9.0

Note: Because the CDH cluster is used, the default version of spark is 1.6.0, and saprk2.3.0 is installed through the parcel package. At this time, there are two spark versions in the cluster. Hue integrates spark 1.6. It is necessary to upload the jar package and oozie-share lib-spark*.jar of spark 2 to share lib of hue. The directory is: / user/oozie/share/lib/lib/lib_201810151907/spark2.

1. Upload jar packages

```
[root@sdw2 jars]# pwd
/opt/cloudera/parcels/SPARK2/lib/spark2/jars
[root@sdw2 jars]# sudo -uhdfs hdfs dfs -put * /user/oozie/share/lib/lib_20181015151907/spark2
[root@sdw2 jars]# cd /opt/cloudera/parcels/CDH/lib/oozie/oozie-sharelib-yarn/lib/spark/
[root@sdw2 spark]# pwd
/opt/cloudera/parcels/CDH/lib/oozie/oozie-sharelib-yarn/lib/spark
[root@sdw2 spark]# sudo -uhdfs hdfs dfs -put oozie-sharelib-spark*.jar /user/oozie/share/lib/lib
```

2. Modification of ownership and authority

```
[root@sdw2 spark]# sudo -uhdfs hdfs dfs -chown -R oozie:oozie /user/oozie/share/lib/lib_201810151
[root@sdw2 spark]# sudo -uhdfs hdfs dfs -chmod -R 775 /user/oozie/share/lib/lib_20181015151907/s
```

3. Update sharelib

```
[root@sdw1 init.d]# oozie  admin --oozie http://dw-greenplum-2:11000/oozie/ --sharelibupdate
[ShareLib update status]
        sharelibDirOld = hdfs://dw-greenplum-2:8020/user/oozie/share/lib/lib_20181015151907
        host = http://dw-greenplum-2:11000/oozie
        sharelibDirNew = hdfs://dw-greenplum-2:8020/user/oozie/share/lib/lib_20181015151907
        status = Successful


[root@sdw1 init.d]# oozie  admin --oozie http://dw-greenplum-2:11000/oozie/ --shareliblist
[Available ShareLib]
hive
spark2
distcp
mapreduce-streaming
spark
oozie
hcatalog
hive2
sqoop
pig
```

II. Problem Description

Description: On HDFS, there is an order data order.txt file, the splitting symbol of the file field ",", and the sample data is as

Hot Keywords

- Java - 5220
- Attribute - 2418
- Programming - 2384
- Database - 2349
- Python - 1973
- xml - 1959
- Javascript - 1902
- Android - 1890
- Spring - 1799
- JSON - 1741
- github - 1704
- network - 1679
- less - 1673
- Linux - 1420
- MySQL - 1315
- PHP - 1267
- SQL - 1211
- encoding - 1179
- Mobile - 1029
- Apache - 896

follows:

Order_00001,Pdt_01,222.8
Order_00001,Pdt_05,25.8
Order_00002,Pdt_03,522.8
Order_00002,Pdt_04,122.4
Order_00002,Pdt_05,722.4
Order_00003,Pdt_01,222.8

The fields in turn represent the order id, the commodity id, and the transaction volume.

Question: Use sparkcore to find the id of the commodity with the largest turnover in each order, and save the result in the hit table.

Three, code

```
package com.company.sparkcore

import org.apache.spark.sql.Row
import org.apache.spark.sql.hive.HiveContext
import org.apache.spark.sql.types.{StringType, StructField, StructType}
import org.apache.spark.{SparkConf, SparkContext}

object TopOrderItemCluster {
  def main(args: Array[String]): Unit = {
    val conf = new SparkConf().setAppName("top n order and item")
    val sc = new SparkContext(conf)
    val hctx = new HiveContext(sc)
//The directory of data on HDFS is: / user/hdfs/spark_data/data.txt
    val orderData = sc.textFile("spark_data/data.txt")
    val splitOrderData = orderData.map(_.split(","))
    val mapOrderData = splitOrderData.map { arrValue =>
      val orderID = arrValue(0)
      val itemID = arrValue(1)
      val total = arrValue(2).toDouble
      (orderID, (itemID, total))
    }
    val groupOrderData = mapOrderData.groupByKey()
    //groupOrderData.foreach(x => println(x))
    //    (Order_00003,CompactBuffer((Pdt_01,222.8)))
    //    (Order_00002,CompactBuffer((Pdt_03,522.8), (Pdt_04,122.4), (Pdt_05,722.4)))
    //    (Order_00001,CompactBuffer((Pdt_01,222.8), (Pdt_05,25.8)))
    val topOrderData = groupOrderData.map(tupleData => {
      val orderid = tupleData._1
      val maxTotal = tupleData._2.toArray.sortWith(_._2 > _._2).take(1)
      (orderid, maxTotal)
    }
    )
    topOrderData.foreach(value =>
      println("Maximum turnover order ID For:" + value._1 + " ,Corresponding commodities ID For:
      //    The order ID of the maximum turnover is Order_00003, and the corresponding commodity
      //    The order ID of the maximum turnover is Order_00002, and the corresponding commodity
      //    The order ID of the maximum turnover is Order_00001, and the corresponding commodity
    )
    //Constructing RDD with metadata Row
    val RowOrderData = topOrderData.map(value => Row(value._1, value._2(0)._1))
    //Building metadata
    val structType = StructType(Array(
      StructField("orderid", StringType, false),
```

```
      StructField("itemid", StringType, false))
    )
    //Converting to DataFrame
    val orderDataDF = hctx.createDataFrame(RowOrderData, structType)

    orderDataDF.registerTempTable("tmptable")
    hctx.sql("CREATE TABLE IF NOT EXISTS orderid_itemid(orderid STRING,itemid STRING) ROW FORMAT
    hctx.sql("insert into orderid_itemid select * from tmptable")
  }

}
```

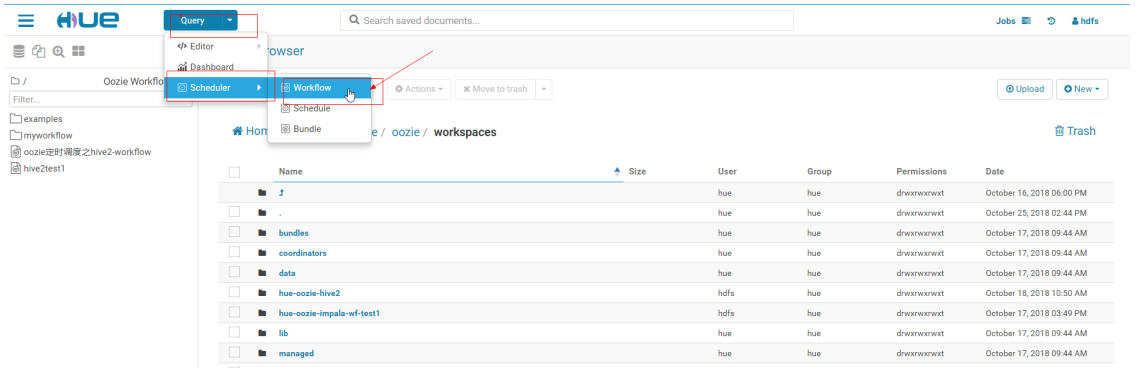
4. Running on Clusters

Place the jar packages on the cluster, test and submit spark jobs

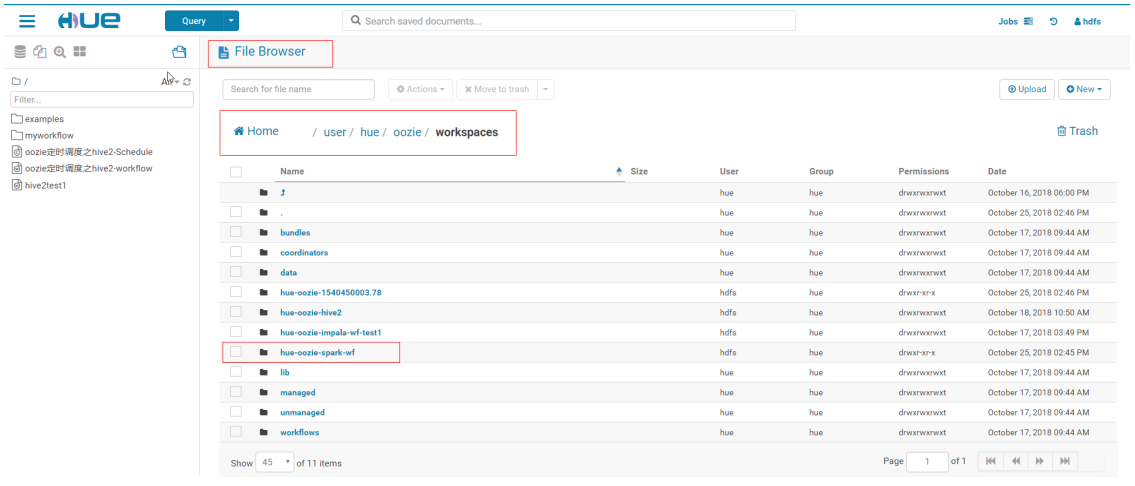
```
##The submit script is submit1.sh
spark2-submit \
--class com.yeexun.sparkcore.TopOrderItemCluster \
--master yarn \
--deploy-mode cluster \
/opt/software/myspark-1.0-SNAPSHOT.jar
```

5. Creating spark workflow through hue

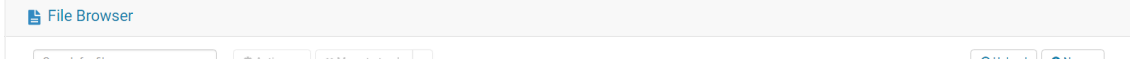
1. Create workflow

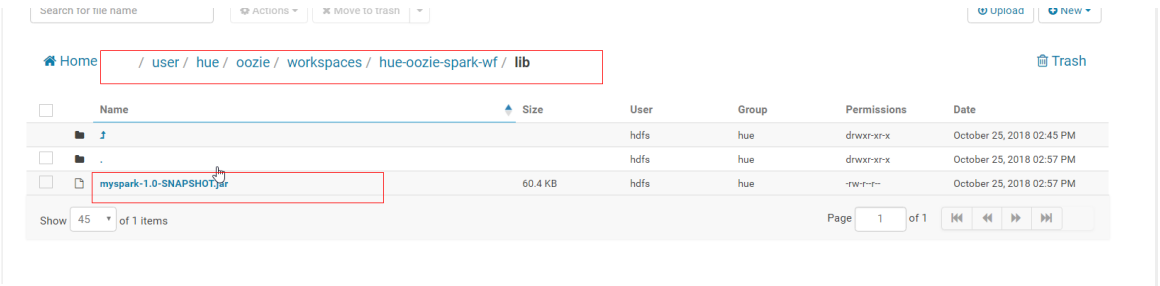


2. After clicking workflow, a folder named hue-oozie-*****. ** will be created automatically under the / user/hue/oozie/workspaces folder. There is a lib folder under the folder. Modify the name of the folder to be hue-oozie-spark-wf (you can modify it or not)

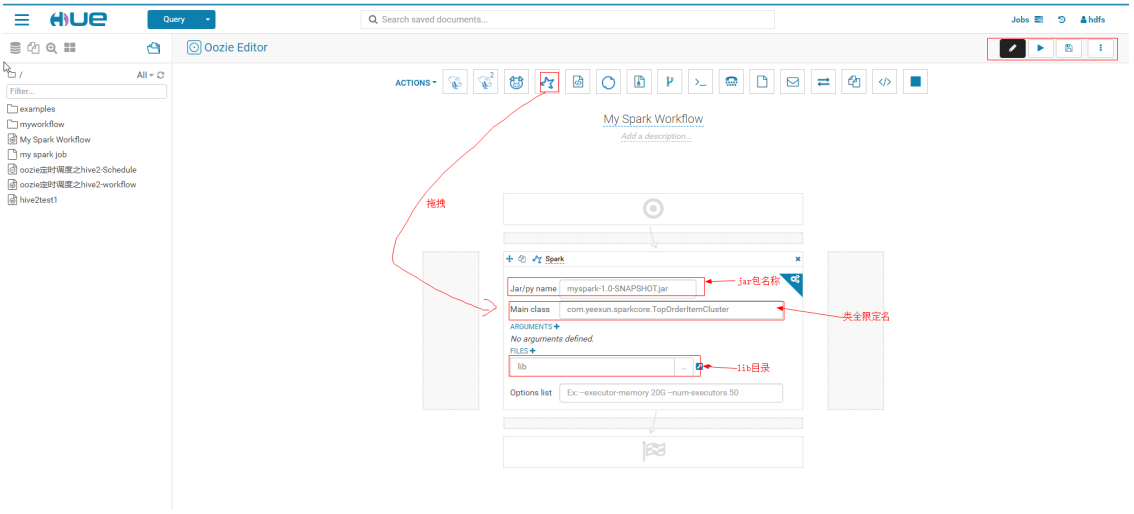


3. Upload the packaged jar package to the / user/hue/oozie/workspaces/hue-oozie-spark-wf/lib folder

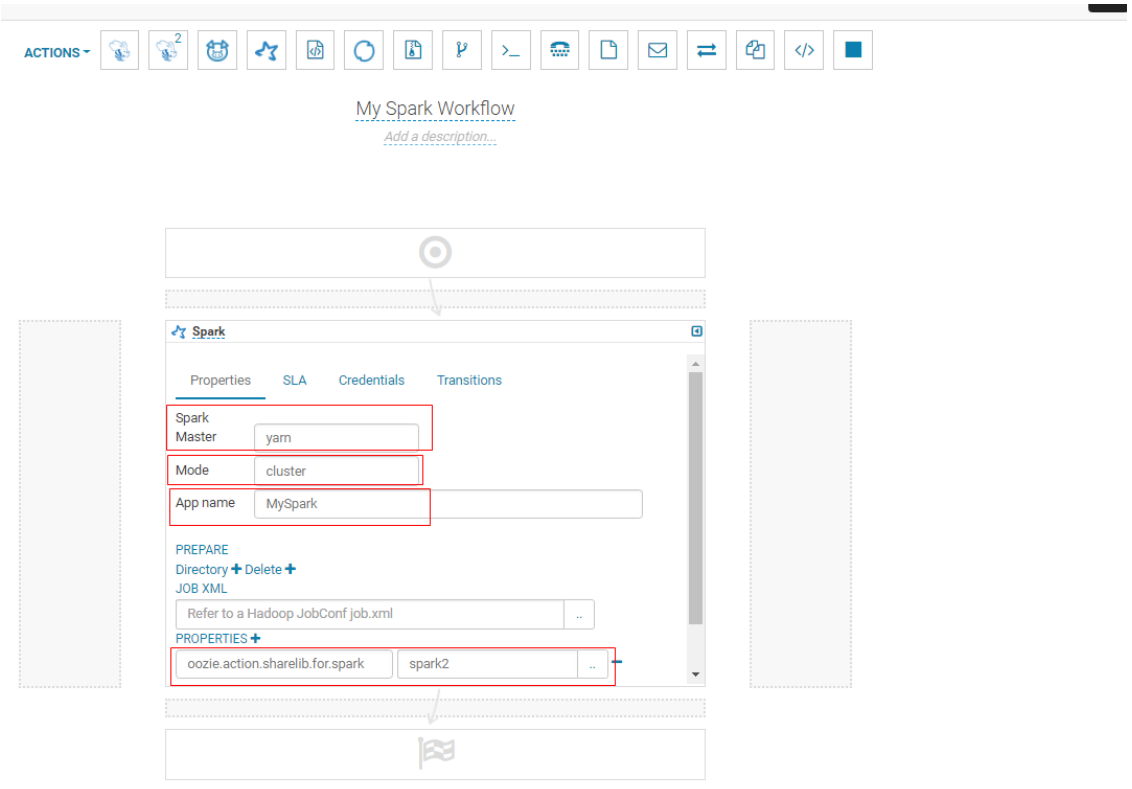




4. Create workflow for spark



5. Click Configuration of editing workflow



6. Click on the top right corner Running workflow

Posted by **Xorandnotor** on *Thu, 24 Jan 2019 10:45:13 -0800*