Assignment - 25.1

Task 1: As discussed in class integrate Spark Hive

Answer:

1. Open acadgild VM and start all daemons – Hadoop + Hbase using below commands.
   - Start-all.sh
   - Start-hbase.sh

2. Now open Eclipse in VM and create a project named "TestScalaProject" and import the source code "SparkHiveTest.scala"

SparkHiveTest.scala

```
1  import org.apache.spark.sql.SparkSession
2
3
4  object SparkHiveTest {
5
6    def main (args: Array[String]) : Unit  = {
7
8      val sparkSession = SparkSession.builder.master("local")
9      .appName("spark session example")
10     .config("spark.sql.warehouse.dir","/user/hive/warehouse")
11     .config("hive.metastore.uris", "thrift://localhost:9083")
12     .enableHiveSupport().getOrCreate()
13     val listOfDB = sparkSession.sqlContext.sql("show databases")
14     listOfDB.show(8,false)
15     println("test");
16   }
17 }
```

3. Resolve all the compilation errors from external JARs which are required and then execute the class as a Scala Application.

4. Copy hive-site.xml file from $HIVE_HOME/conf to $SPARK_HOME/conf.

5. Add the following property to hive-site.xml on Spark side:

```
<property>
 <name>hive.metastore.uris</name>
 <value>thrift://localhost:9083</value>
 <description>password for connecting to mysql server</description>
</property>
```

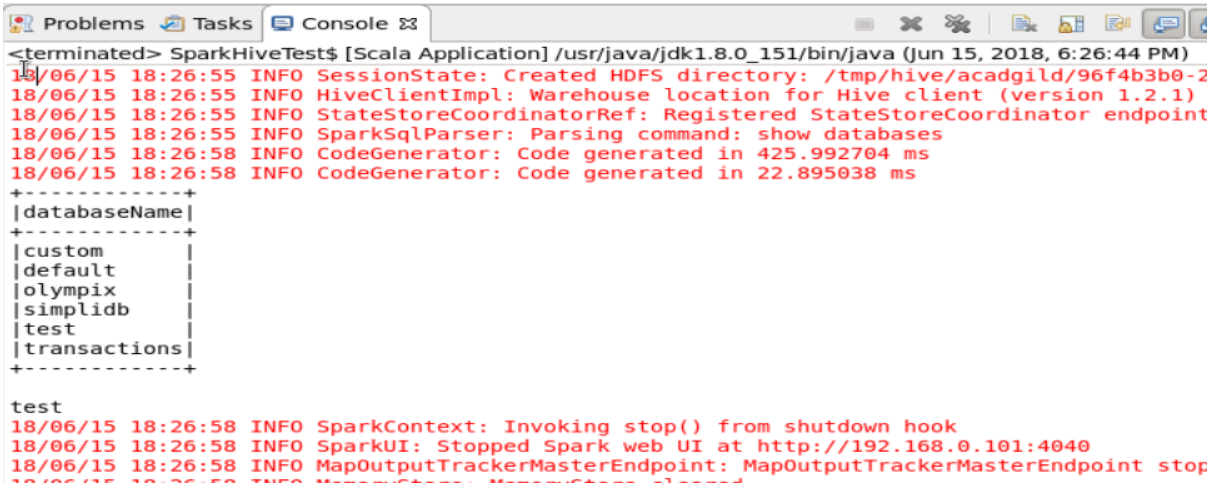
   This property helps to create the connection between Spark and Hive.

6. Start hive metastore using the command – "hive –service metastore"

```
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ hive --service metastore
```

Make sure mysql service is running or else execute the command “sudo service mysqld start”.

7. Run the class in eclipse as “Scala Application”.



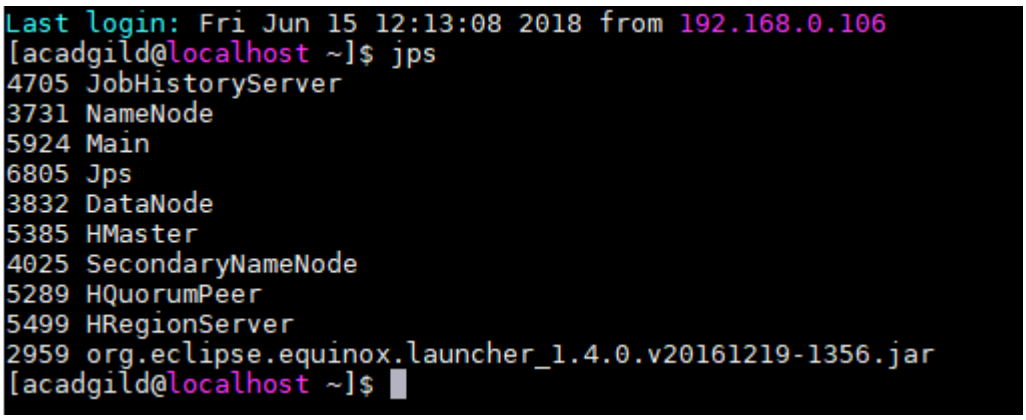
Here we could see the list of database in Hive using Apache Spark – Hadoop Integration.

Task 2: As discussed in class integrate Spark Hbase

Answer: Follow the below steps to proceed for Spark Hbase integration

8. Open acadgild VM and start all daemons – Hadoop + Hbase using below commands.
   - Start-all.sh
   - Start-hbase.sh
   - Mr-jobhistory-daemon.sh start history server

   Check all daemons are running as in below screenshot.



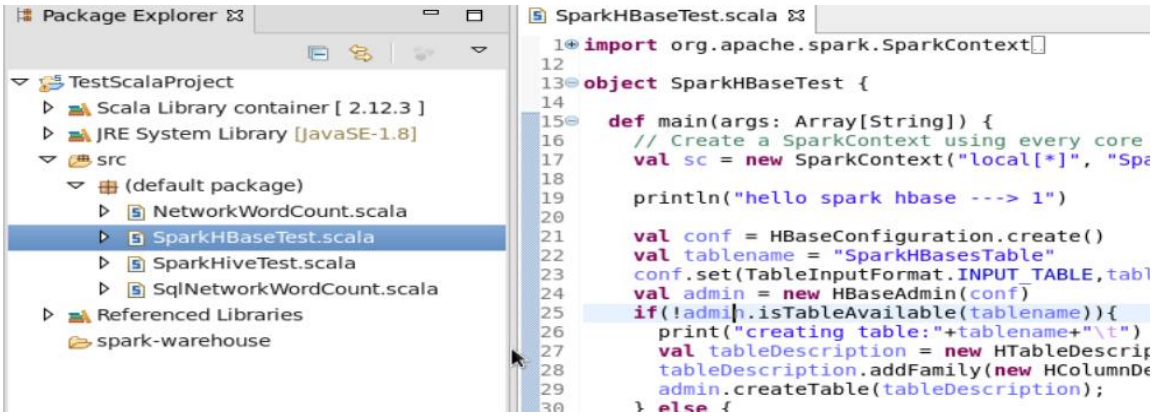
9. Open HBase shell and execute the list command.

```
hbase(main):001:0> list
TABLE
SparkHBasesTable
txn_count
2 row(s) in 0.4380 seconds

=> ["SparkHBasesTable", "txn_count"]
hbase(main):002:0>
```

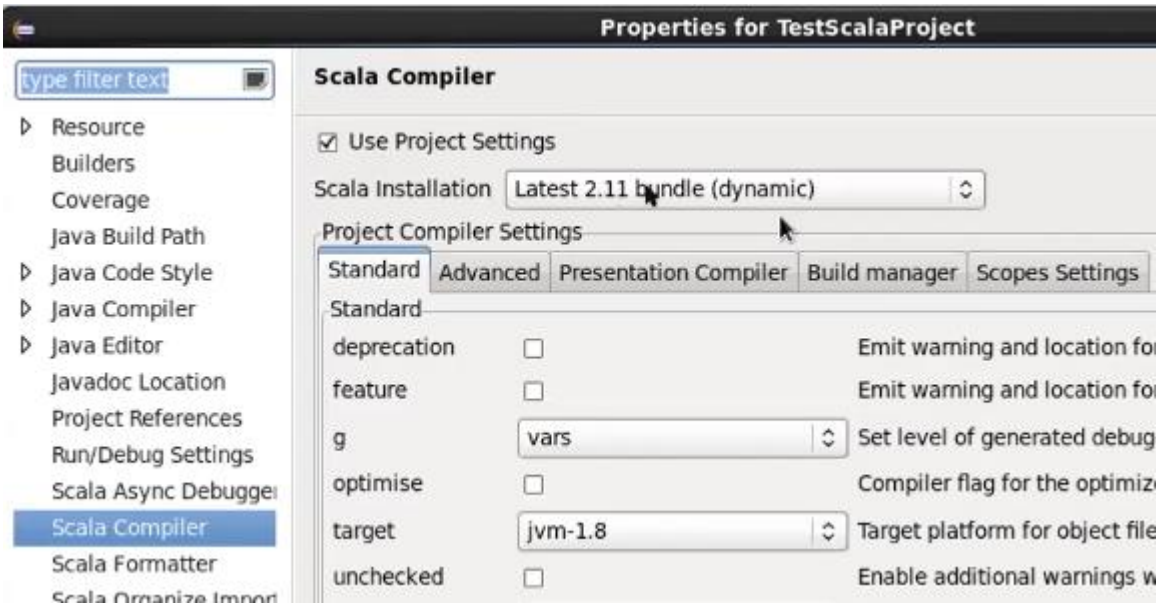

10. Disable and drop the SparkHBasesTable.

```
hbase(main):003:0> disable 'SparkHBasesTable'
0 row(s) in 2.4430 seconds

hbase(main):004:0> drop 'SparkHBasesTable'
0 row(s) in 1.2890 seconds

hbase(main):005:0>
```

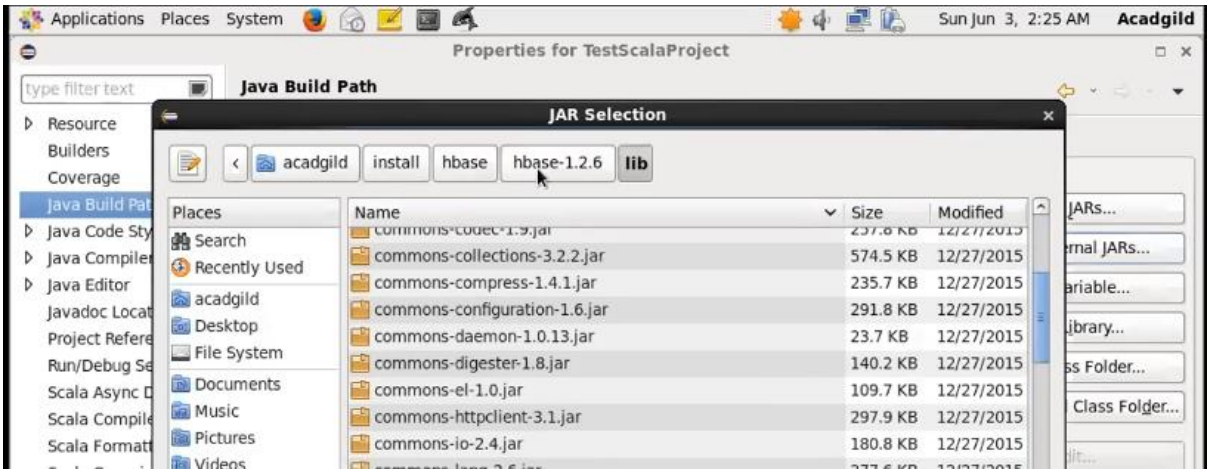
11. Now open Eclipse in VM and create a project named “TestScalaProject” and import the source code “SparkHBaseTest.scala”



12. Reslove all the error by adding on the external JARs from Spark folder and make sure the Scala compiler version is “Latest 2.11 bundle (dynamic)”.



JAR details from this path:

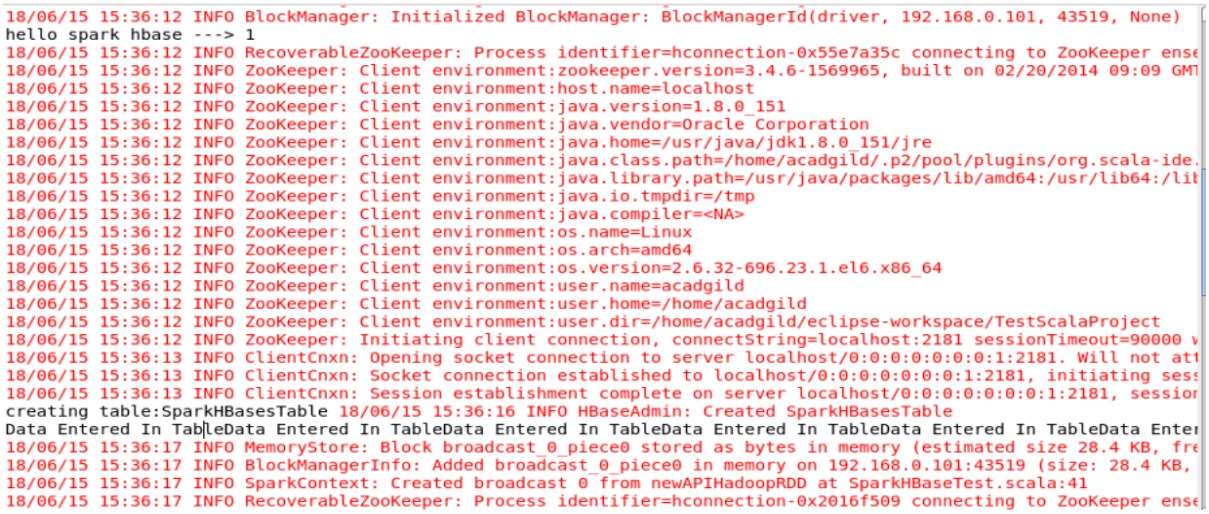


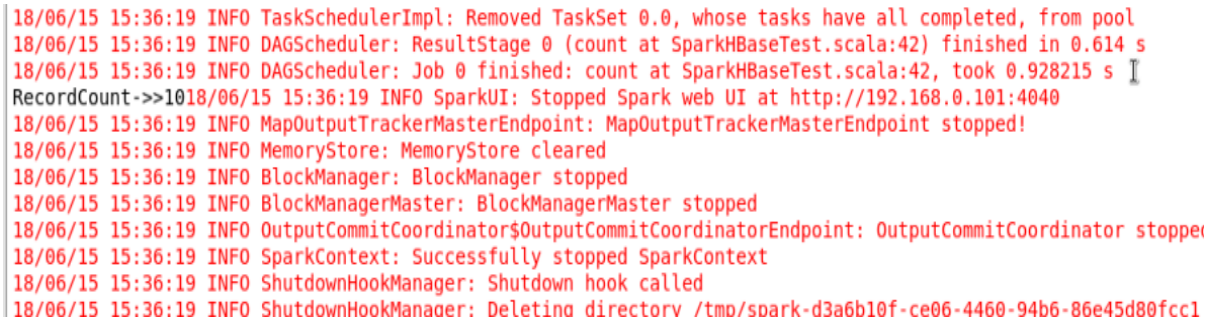
13. Now run the particular class “SparkHBaseTest.scala” as Scala application.

*SparkHBaseTest.scala

```
15  def main(args: Array[String]) {
16    // Create a SparkContext using every core of the local machine, named RatingsCounter
17    val sc = new SparkContext("local[*]", "SparkHBaseTest")
18
19    println("hello spark hbase ---> 1")
20
21    val conf = HBaseConfiguration.create()
22    val tablename = "SparkHBasesTable"
23    conf.set(TableInputFormat.INPUT_TABLE,tablename)
24    val admin = new HBaseAdmin(conf)
25    if(!admin.isTableAvailable(tablename)){
26      print("creating table:"+tablename+"\t")
27      val tableDescription = new HTableDescriptor(tablename)
28      tableDescription.addFamily(new HColumnDescriptor("cf".getBytes()));
29      admin.createTable(tableDescription);
30    } else {
31      print("table already exists")
32    }
33
34    val table = new HTable(conf,tablename);
35    for(x <- 1 to 10){
36      var p = new Put(new String("row" + x).getBytes());
37      p.add("cf".getBytes(),"column1".getBytes(),new String("value" + x).getBytes());
38      table.put(p);
39      print("Data Entered In Table")
40    }
41    val hBaseRDD = sc.newAPIHadoopRDD(conf, classOf[TableInputFormat],
42        classOf[ImmutableBytesWritable],classOf[Result])
43    print("RecordCount->>"+hBaseRDD.count())
44    sc.stop()
```

14. Above code – creates a new table and enters input values to the table in HBase using Spark API. Below are the output screens for the same.





Here we have got the output in the eclipse console.

Now check the same in the terminal using hbase shell.