cs523 - Big Data Technology

Lab 9 – Apache Spark

Submit your *own work* on time. No credit will be given if the lab is submitted after the due date.
Follow the instructions completely.

Step 1: Verify if Java is installed

Java is a pre-requisite software for running Spark Applications.
Use the following command to verify if Java is installed on your system :-

```
[cloudera@quickstart ~]$ java -version
java version "1.8.0_211"
Java(TM) SE Runtime Environment (build 1.8.0_211-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.211-b12, mixed mode)
```

Support for Java 7 is deprecated as of Spark 2.0.0

Refer to "InstallJdk8inCDH.pdf" documentation to upgrade your CDH to JDK8.

1. Spark Practice Lab

No need to submit this part.

I. Spark WordCount in Local mode
II. Spark WordCount in Pseudo-distributed mode
III. Spark Shell example

2. Spark Homework Lab

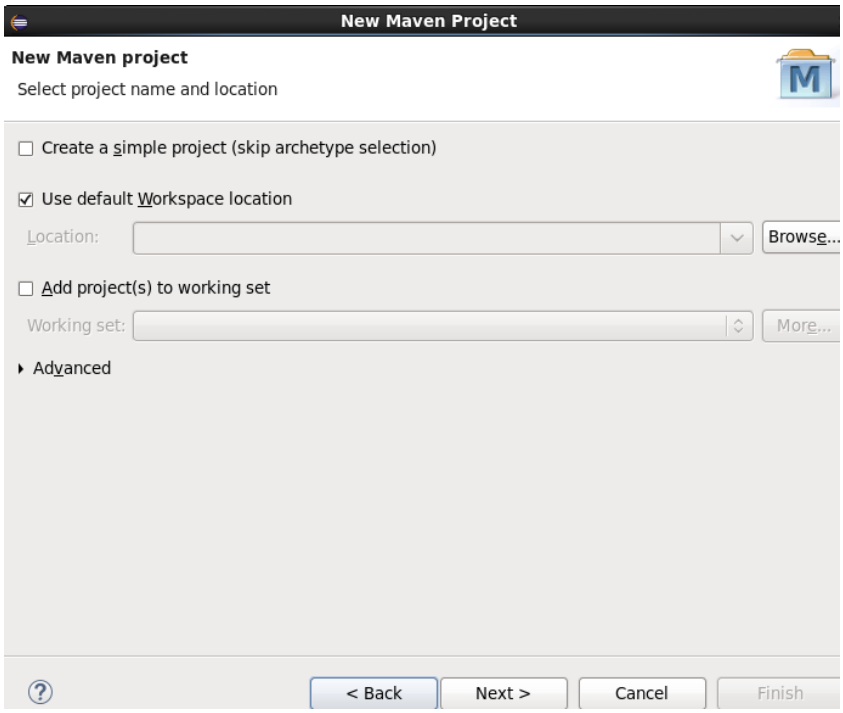
Submit ".java" files and output along with the command that you used to run the program in pseudo-distributed mode.

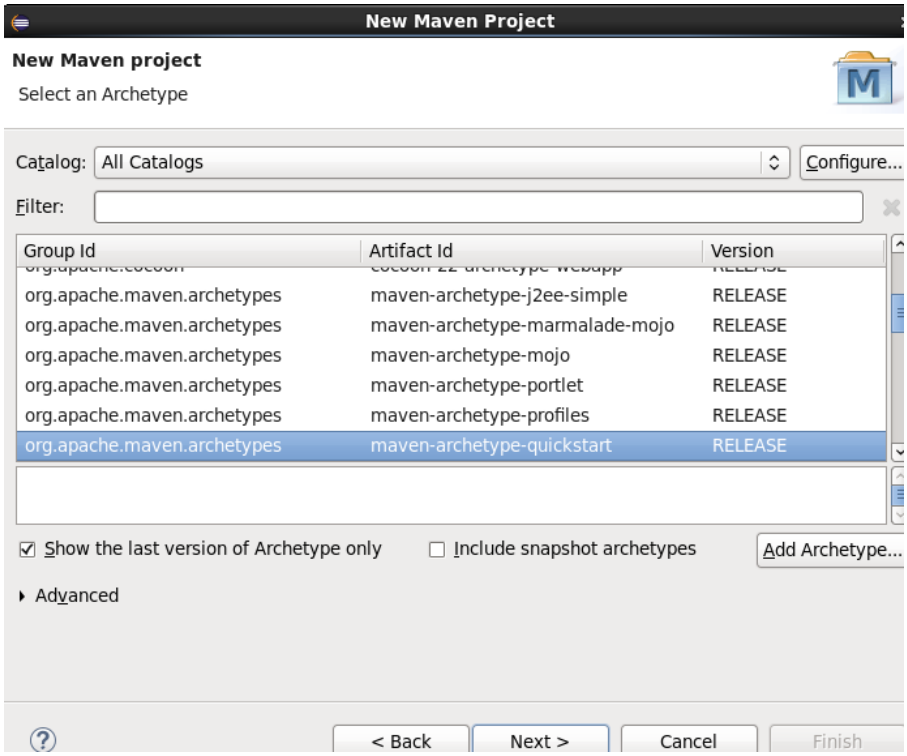
Paste screenshots wherever applicable.

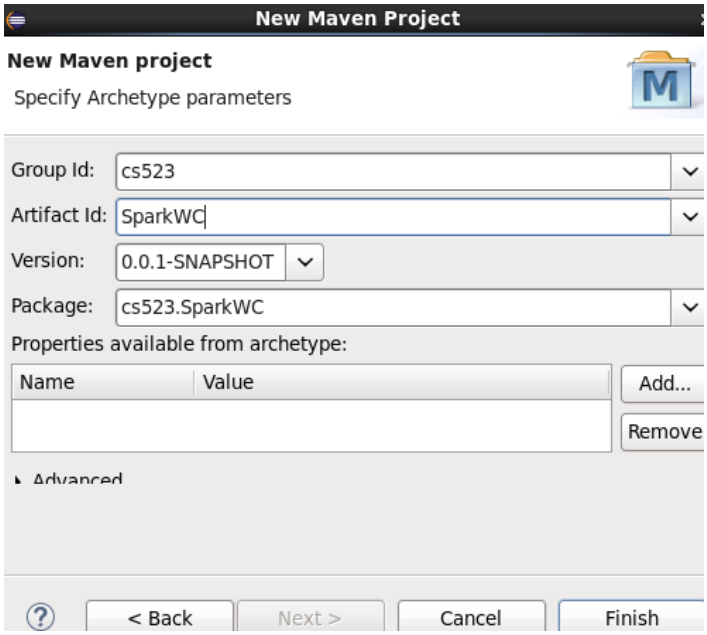
Spark Practice Labs

Part 1 - Spark Word Count in local mode

1. Need to create a Maven project in Eclipse for spark word count.







2. Spark Word Count Java programs (JDK 7 and 8) are given to you, download them and add them to your project. Create two separate Maven projects for jdk7 and jdk8 programs. (You can safely delete App.java and test packages that got created by default)
3. Your pom.xml file should look like the one which is given to you. It'll add all the required dependencies to your project.
4. Create folder "input" in your project path and add a test input file there on which you'll run the word count.
5. Run the given word count java programs as Java application.
6. You can check the part file in the output folder after refreshing the project.

Part 2 - Spark Word Count in pseudo-distributed mode

---

1. Make sure your input folder is present under "/user/cloudera" directory and it has the input file in it.
2. Create jar of your project using *mvn package* command or from eclipse using *maven build* and keep the *goal* as *package*. The created jar file will be inside the target directory. I'm saving the jar file to Desktop but you can choose any other path.
3. Now you need to submit this jar to the Spark cluster which is running on YARN using the following command.
   spark-submit --class "cs523.SparkWCjdk8.SparkWordCountjdk8" --master yarn Desktop/SparkWCjdk8.jar /user/cloudera/input/ /user/cloudera/output
4. After successful execution, it'll create an output folder in "/user/cloudera". Check it and verify the part file.

Part 3 - Spark Shell Example

In this part, you'll start the Spark shell (spark-shell) and use it to write a Spark example program in Scala.
In spark shell, you can use *CTRL-L* to clear the screen and *exit* for coming out of the shell.

Problem Statement: Spark uses some third-party libraries. You need to find out how many of these are licensed under the BSD license (acronym for Berkeley Software Distribution).

Luckily, Spark comes with a file named LICENSE, located in the Spark root directory. The LICENSE file contains the list of all libraries used by Spark and the licenses they're provided under. Lines in the file, which names packages licensed under the BSD license, contain the word BSD. You could easily use a Linux shell command to count those lines, but that's not the point.

Let's see how to count the lines using the Scala Spark API:

```
scala> val licLines = sc.textFile("file:///usr/lib/spark/LICENSE")

licLines: org.apache.spark.rdd.RDD[String] = file:///usr/lib/spark/LICENSE
MapPartitionsRDD[5] at textFile at <console>:27

scala> val lineCnt = licLines.count

lineCnt: Long = 294

scala> val bsdLines = licLines.filter(line => line.contains("BSD"))

bsdLines: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at filter at
<console>:23

scala> bsdLines.count

res0: Long = 34
```

Spark Homework Lab

Complete this HW using Java 8.

1. This homework is an enhanced version of WordCount. In this version of WordCount, the goal is to learn the distribution of letters in the most popular words in a corpus; meaning now we need to calculate **character (letter) count *of the most popular words***.

   Your application must do the following:

   a) Gets a word frequency threshold from user.

   b) Reads an input set of text documents.

   c) Counts the number of times each word appears.

   d) Filters out all words that appear fewer times than the threshold.

   e) For the remaining words, counts the number of times each letter occurs.

[Optional Question – 3 bonus points]

2. **Analysis on Apache log file**

   a) Get the Apache log file samples from here.

   b) Find Apache log file format from here and refer to this link for parsing the file.
   Note that these log file formats keep changing and the Regex that is available in the above link might be a little different. So, figure it out yourself!

   c) Find out how many 401 errors are there within a particular date range which will be supplied by the user.

   d) List all the IP addresses that have accessed this server more than 20 times.