cs523 - Big Data Technology

Lab 9 - Spark

Submit your *own work* on time. No credit will be given if the assignment 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 -

```
$java -version
java version "1.7.0_67"
Java(TM) SE Runtime Environment (build 1.7.0_67-b01)
Java HotSpot(TM) 64-Bit Server VM (build 24.65-b04, mixed mode)
```

Support for Java 7 is deprecated as of Spark 2.0.0

Refer to “InstallJdk8inCDH.pdf” documentation to upgrade your quick start VM 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

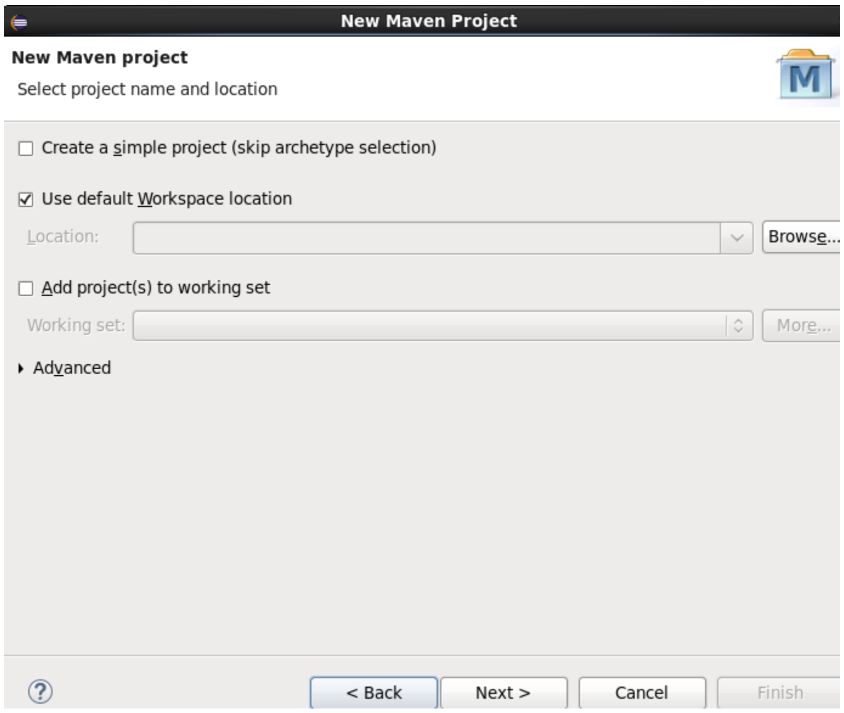
2. Spark Homework Lab

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

Paste screenshots wherever applicable.

Part 1 - Spark Word Count in local mode

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



New Maven Project

New Maven project

Select project name and location

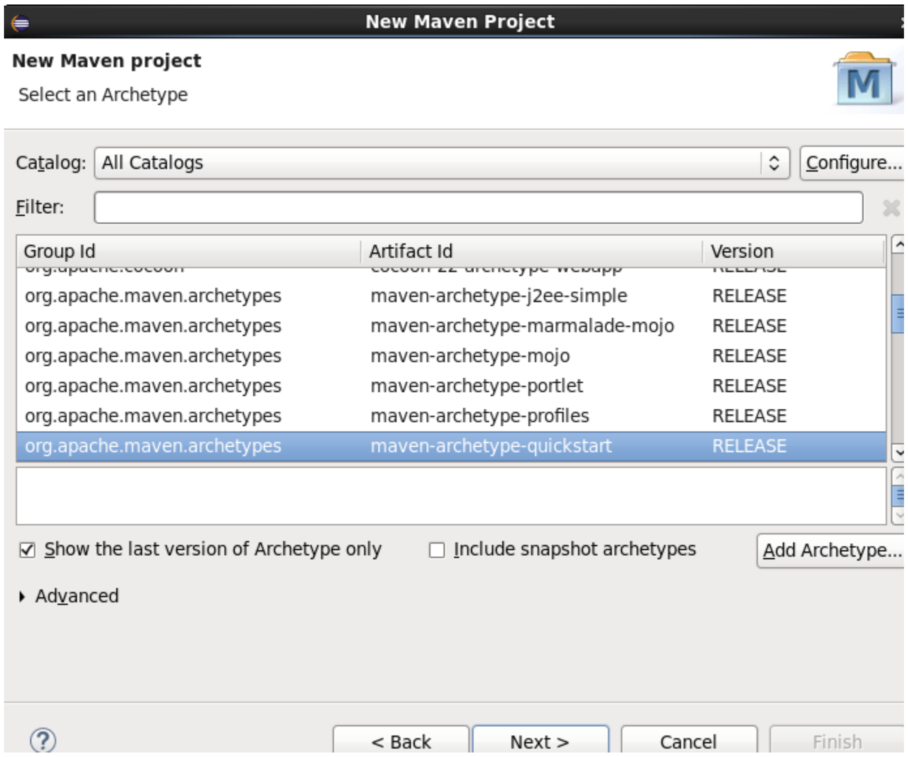
☐ Create a simple project (skip archetype selection)

☑ Use default Workspace location

Location:

☐ Add project(s) to working set

Working set:



New Maven Project

New Maven project

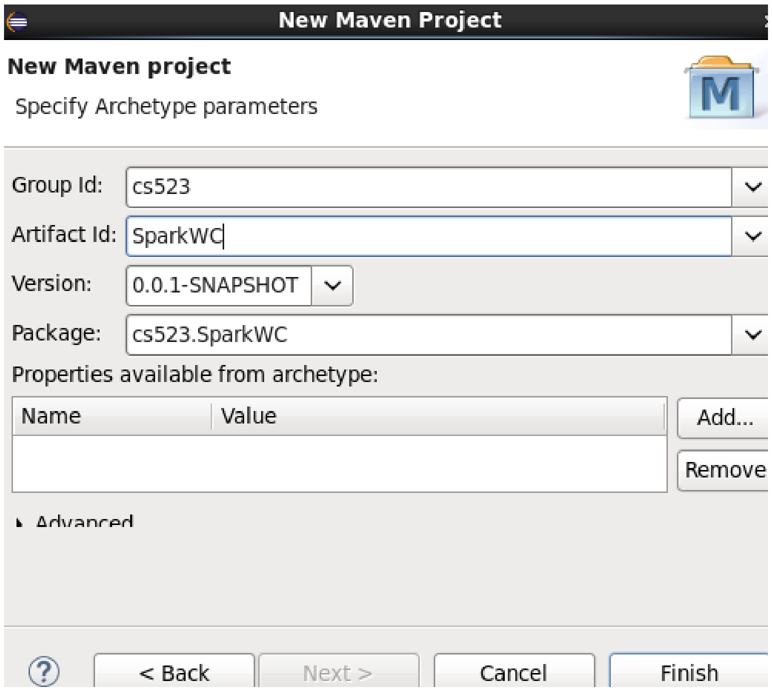
Select an Archetype

Catalog: All Catalogs

Filter:

Group Id	Artifact Id	Version
org.apache.maven.archetypes	maven-archetype-j2ee-simple	RELEASE
org.apache.maven.archetypes	maven-archetype-marmalade-mojo	RELEASE
org.apache.maven.archetypes	maven-archetype-mojo	RELEASE
org.apache.maven.archetypes	maven-archetype-portlet	RELEASE
org.apache.maven.archetypes	maven-archetype-profiles	RELEASE
org.apache.maven.archetypes	maven-archetype-quickstart	RELEASE

☑ Show the last version of Archetype only ☐ Include snapshot archetypes


New Maven project

Specify Archetype parameters

Group Id: cs523

Artifact Id: SparkWC

Version: 0.0.1-SNAPSHOT

Package: cs523.SparkWC

Properties available from archetype:

Name	Value

2. Spark Word Count Java programs (JDK 7 and 8) are given to you, download them and add it to your project. (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 and add a test input file there on which you'll use to run the word count.
5. Run the WordCount.java program 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. Export the jar of your project. 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.SparkWC.WordCountjdk8" --master local
   Desktop/SparkWC.jar /user/cloudera/input/input.txt /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

The HW should be completed 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.

   ----

2. **Analysis on Apache log file.**
   a) Find Apache log file format.
   b) Get the Apache log file samples from here**.**
   c) Find out how many 200 errors are there within a particular date range.
   d) List all the IP addresses that have accessed this server more than 10 times.

Working with Databricks cloud

https://www.youtube.com/embed/MXI0F8zfKGI