Supercomputing for Big Data ET4310 (2016)

Assignment 1 (Using Spark for In-Memory Computation)

Hamid Mushtaq and Zaid Al-Ars

Introduction

Spark is a fast and powerful big data engine for processing Hadoop data. It runs in Hadoop clusters through Hadoop YARN or in Spark's standalone mode, and it can process data in HDFS, HBase, Hive, and any Hadoop input format. It is designed to perform both general data processing (similar to MapReduce) and new workloads like streaming, interactive queries, and machine learning.

In this assignment you'll learn about Spark, how to use it and the tools and facilities it provides for processing your big data analytics. You'll be introduced to a set of Spark commands and you'll experiment with the processing they are capable of. You'll also write and execute a twitter based stream processing application using Spark.

System requirements and Installation

Since Spark is java based, it is possible to use it with any major operating system. However, since it is most commonly used with Linux, we will recommend you to use Linux. To use Spark, you must have the following software packages on your computer.

- Java version 1.7
- Spark 2.0.0 (spark-2.0.0-bin-hadoop2.7.tgz)
- sbt (https://dl.bintray.com/sbt/native-packages/sbt/0.13.12/sbt-0.13.12.tgz)

For Spark, you must set the SPARK_HOME environment variable. Assuming that you put the spark folder in your home directory, you would set the SPARK_HOME variable using the following line.

export SPARK_HOME=~/spark-2.0.0-bin-hadoop2.7

Moreover, you must add the bin folder of sbt to your PATH environment variable. Assuming that you put the sbt folder in your home directory, you would have the following line to do so.

export PATH=$PATH:~/sbt/bin

If you use the templates that we provide with this assignment, with these settings, you can just compile your code by typing **sbt assembly**. Moreover, you could run the compiled jar file using the **run.sh** script provided with those templates.

Note that compiling code with **sbt** could generate a lot of intermediate files, which could take up to 1 GB of hard disk space. So, please make sure that you have enough disk space on the pc where you are compiling your code.

Important points

You must keep the following points in mind for this assignment.

- The division of marks is the following
  - Code functionality (40%)
  - Code (Readability, structure, performance, etc.) (20%)
  - Comments (20%)
  - Report (20%)
- Each statement in your code must be accompanied by a comment, especially on each RDD transformation or action. This is so that you could demonstrate that you thoroughly understand your code, even if you took a bit of help from somewhere. For each RDD, you must at least specify the type of its elements in your comments. For example

```
// (id, (language, retweetCount, text))
SomeRDD = …
```

- In your report, you don't need to go too much into details, as through your comments, you should already be able to explain your code quite well. Secondly, the report should be no longer than 5 pages. You must use the following template for the report.

  https://www.dropbox.com/s/m6fm61korjcuwrt/Report.docx?dl=0

  Make sure also that you follow the points given in that template.
- You must submit a folder (in zipped form) containing all the 3 exercises and your report. That zip file should be named as **SBD_L1_YourName_YourStudentNumber**. For all the three exercises, you must use the templates provided for those exercises. You can modify the source files (obviously), run scripts and even the build scripts, and even add more source files if required.

Exercise 1: Data exploration using Spark (25% of the grade)

This exercise is to perform some data exploration of page view statistics for Wikimedia projects. The exercise is somewhat similar to Berkley Spark tutorial to be found at the following link. There, you can get a few hints on how to solve this exercise.

http://ampcamp.berkeley.edu/big-data-mini-course/data-exploration-using-spark.html

You must start with the following template to write your code.

https://www.dropbox.com/sh/1j6yc7r0nvsaedo/AADcoKC6lKhnKokjGBN_pWAUa?dl=0

For input, you can download any Pagecount files for 2016-08. These files can be found at https://dumps.wikimedia.org/other/pagecounts-raw/2016/2016-08/ with .gz extension. For example, one such file is https://dumps.wikimedia.org/other/pagecounts-raw/2016/2016-08/pagecounts-20160801-040000.gz

The schema is <Language code> <Page title> <View count> <Page size>. Each of these fields is delimited by space. More about the format of these files can be found at https://dumps.wikimedia.org/other/pagecounts-raw/

Note that the first field (Language code) can have some text after a dot, for example **fr.b**. We want to only get the language code and ignore the text after the dot. So instead of **fr.b**, we must use **fr** for example. This can be done by using the **org.apache.commons.lang3.StringUtils** library. To get the text before the dot, we can simply call it as **StringUtils.substringBefore(text, ".")**.

Also, you must filter out records, where the Page title is the same as the Language code. For example, if you have a record where the Language code is **fr** and the Page title is also **fr**, filter it out.

The task of this assignment is to output information in the following form.

<Language>,<Language-code>,<TotalViewsInThatLang>,<MostVisitedPageInThatLang>,<ViewsOfThatPage>

So each line has a record for a different language, where you have the name of that language, its code, the combined views of all pages in that language, the title of the most visited page in that language, and also the views of that most visited page in that language.

The records should be sorted such that the language with the most views overall (Views of all pages in that language combined) should be listed first. In other words, records are listed in descending order according to the total number of views of pages in different languages. To make things clearer, you can download a sample output from
https://www.dropbox.com/s/zp6npdxqaviu51w/part1.txt?dl=0

Before starting with this assignment, you might want to go through the following links to get accustomed to Spark and Scala.

http://spark.apache.org/docs/latest/quick-start.html

http://spark.apache.org/docs/latest/programming-guide.html

http://spark.apache.org/docs/latest/tuning.html

Exercise 2: Spark streaming (45% of the grade)

Before starting this assignment, you must first get accustomed to Spark streaming, using the Twitter example which displays the top 10 hash tags, each second. You can download that example from
https://www.dropbox.com/sh/7mewqbyo0hbxcxs/AADqu2B9wcxcqTVt4jBfzL4ca?dl=0

However, note that you will have to fill in properly the values of apiKey, apiSecret, acessToken and accessTokenSecret, before you can run this example. The method on how to do that is explained in
http://ampcamp.berkeley.edu/3/exercises/realtime-processing-with-spark-streaming.html

The code in the example is also explained there. Note however that since that tutorial uses older Spark libraries, the code given there is not exactly the same as the example code we have provided, but quite similar nevertheless.

You can learn more about Spark streaming from the following link.

http://spark.apache.org/docs/latest/streaming-programming-guide.html

For this Exercise, You have to modify the template code given at https://www.dropbox.com/sh/3n56x118rk0px98/AAAxousLkXHd9NXKw5bl-l-ua?dl=0

The task of this exercise is to write a standalone Spark streaming application with the following characteristics.

- Continuously read Twitter feed every 5 seconds.
- Collect the feed in a sliding window of 60 seconds.
- After each 5 seconds, calculate the number of retweets through all the retweets appearing in that window. For this, first you have to filter the tweets, so that you are left only with retweets. From a retweet, you can get the status of the original tweet (http://twitter4j.org/javadoc/twitter4j/Status.html) and therefore also get its retweet count. Note however, that since the free twitter feed gives you only 1% of the tweets, just counting the number of retweets in a window is not a good method. What instead you must do for this assignment is to get the retweet count from the retweet that appeared first in the window (lets call that count value X) and get the retweet count from the retweet that appeared last in the window (lets call that count value Y). Then, using Y – X + 1, you could get the retweet count for the retweeted tweet.
- That data that needs to be logged each 5 seconds has to be of the following format <Secs>,<Lang>,<Langcode>,<TotalRetweetsInThatLang>,<IDOfTweet>,<MaxRetweetCount>,<MinRetweetCount>,<RetweetCount>,<Text>
- You can get the language used in the retweet by using the Apache Tika library (included with the template code). It uses text analytics to figure out the language used. Since, tweets contain short texts and informal language, many times Apache Tika will figure out wrong language, for example in some cases, it will detect English as Norwegian. This kind of analytics can be improved by using Machine learning, but for this assignment, it is enough to use Apache Tika despite its inaccuracies.
- The **TotalRetweetsInThatLang** has to be found out by combining all the retweet counts for tweets in that language**. IDOfTweet** is the ID of the original tweet which has been retweeted. **MaxRetweetCount** is found by looking at the retweet count through the retweet that appeared last in the window (Hint: Reduce by **Math.max** function) while **MinRetweetCount** is found by looking at the retweet count through the retweet that appeared first in the window (Hint: Reduce by **Math.min** function). Adding 1 to the difference between **MaxRetweetCount** and **MinRetweetCount** will give you the **RetweetCount** for that tweet. Finally, **Text** is the text that appears in the tweet.
- When you log data after every 5 seconds, you have to sort it using the **TotalRetweetsInThatLang** value. This means that tweets for the language which has the most combined retweet counts will appear first. Likewise, Tweets for other languages will follow in descending order. For the tweets of the same language, the one with the most retweet count should appear first, and others should follow in descending order.
- To make things clearer, you can look at the sample output at https://www.dropbox.com/s/9vbrytyofalc94g/part2.txt?dl=0

Exercise 3: More data exploration (30% of the grade)

For this assignment, you are going to use the output of Exercise 2 as input. If you have not completed the Exercise 2 when doing this Exercise, you can use the sample output for Exercise 2 to test your code. Otherwise run the program of Exercise 2 for several minutes (say 10 minutes) and use its output as input for this Exercise.

For this Exercise, you have to modify the template code given at https://www.dropbox.com/sh/1p8fodcu7mvqk93/AAAK70fsNJcFQvL06uzZ6prna?dl=0

The task of this exercise is to write a standalone Spark application with the following characteristcs.

- Read the output log of Exercise 2, and calculate the retweet counts for the tweets. For counting the retweet counts, you have to search for the retweets in the whole log for **MinRetweetCounts** and **MaxRetweetCounts**, to see the mimimum value of **MinRetweetCounts** and the maximum value of **MaxRetweetCounts**, and calculate the retweet count through them. This can be better explained by an example. Let us say, we have a tweet X, whose minimum **MinRetweetCounts** value can be found at 75th second, and that minimum value is 64, while its maximum **MaxRetweetCounts** values is found at 380th second and that maximum value is 364, then the **RetweetCount** for that tweet is 364 – 64 + 1 = 301.
- When you print the output, just like in Exercise 2, you have to group the tweets by Language, and sort the tweets by the total retweet counts (combined sum of all the retweet counts) in those languages. This means that tweets for the language with the most retweet counts would appear first, and others would follow in descending order. For the tweets of the same language, the one with the most retweet count would appear first. Likewise, others would follow in descending order.
- You should not show the tweets which have been retweeted less than 2 times. Basically, you just have to filter them out. However, when calculating the value of **TotalRetweetsInThatLang**, you must also consider retweets who have been tweeted only once.
- The format of the output should be the following,

  `Language,Languagecode,TotalRetweetsInThatLang,IDOfTweet,RetweetCount,Text`
- To make things clearer, you can download the sample output (generated by using the sample output of Exercise 2 as input) from https://www.dropbox.com/s/bnsgpm9m03anz6f/part3.txt?dl=0