INFSCI 2750: Cloud Computing

Mini Project 2

March 18, 2018

Objective

The objective of this mini project is to get familiarized with Apache Spark. This project can be done in groups of 2 to 3 members and you will be working on the previously assigned VM's.

Part 1: Setting up Spark: (40 points)

The first task is to configure Spark distribution on top of the previously installed Hadoop cluster. Your setup should make use of YARN for scheduling/running Spark applications. The entire Spark setup should be configured on top of a two or three node Hadoop cluster. In case if your team has only one working VM, drop us an e-mail so that we can assign you a new additional VM to work with.

You can go through the following link for deploying Spark in a standalone mode:
http://spark.apache.org/docs/latest/spark-standalone.html

Once you successfully set up Spark in standalone mode, the next step is to integrate it with the YARN manager.

Here are a few tutorials for integrating Spark with Yarn –
http://spark.apache.org/docs/latest/running-on-yarn.html

Part 2: Developing Spark programs (30 points)

As part of the project you will be working with the '*hetrec2011-lastfm-2k*' data set. In particular you will be working on the *'user_artists.dat'* data file.

The task is to printout the total listening counts of each artist. Your program should print out the listening counts per artist in descending order. In the 'user_artists.dat' file, listening counts for each user-artist pair is indicated by the variable 'weight'.

More details of the dataset could be found here –
http://grouplens.org/datasets/hetrec-2011/

Here are some examples for programming with Spark:
http://spark.apache.org/docs/latest/quick-start.html

Part 3: Developing Spark programs 2 (30 points)

As part of the project you will be working with the log data set which is provided in access_log.zip in the Mini Project 1.

In this project, you need to program a program using Spark to answer the questions and also provide the performance measurements for whether processing the data with cached RDD (resilient distributed dataset: http://spark.apache.org/docs/latest/rdd-programming-guide.html#resilient-distributed-datasets-rdds) or not.

You need to firstly answer the following questions:

1. How many hits were made to the website item "/assets/img/loading.gif"?
2. How many hits were made to the website item "/assets/js/lightbox.js"?
3. Which path in the website has been hit most? How many hits were made to the path?
4. Which IP accesses the website most? How many accesses were made by it?

You can compare the question 3 and 4 with the results you got in Mini Project 1 to verify them.

The second task for Part 3 is to report the performance measurement (running time of the program) of using cached RDD or not use cached RDD. RDD is the most important improvement for Spark compared with Hadoop. To know how to appropriately use the features of it is a must know for playing with Spark.

You can try to reuse some intermediate results in Spark to test it. For example: Here are two ways to get the answer of question 1 and 2:

1. For calculating the hits number to "/assets/img/loading.gif" and "/assets/js/lightbox.js", you can calculate the result directly by counting the object "/assets/img/loading.gif" and "/assets/js/lightbox.js" separately from the whole file (For **each** key searching load the file **once** and fully searching the file to get the answer, so the file needs to be loaded twice and transferred into RDD) or
2. You can reuse the intermediate result of counting all the objects then just output the result of object "/assets/img/loading.gif" and "/assets/js/lightbox.js" (load the file **once** with counting **all** the objects, so the file is loaded only once and transferred into RDD).

The submission should include the outputs (screen shot) of the performance measurement (running time) of the above example.

Project Submission: Submit a **single ZIP file** with your *Pitt email ID* as its filename via the CourseWeb system. The package should contain all your source files and a *readme* file that explains how to execute your program. Also include screenshots of your programs output and spark shell. The IP address of the master machine should be clearly visible in the screenshots. In addition, for Part3, the *readme* file needs include the screenshots of the calculating time of the two conditions: i) the data file is first initialed to a RDD object and you output the calculating time of one program. ii) the data are already processed