CS 6240: Assignment 4

Goal: Implement PageRank in Spark to compare the behavior of an iterative graph algorithm in Spark vs. Hadoop MapReduce.

This homework is to be completed individually (i.e., no teams). You have to create all deliverables yourself from scratch. In particular, it is not allowed to copy someone else's code or text and modify it. (If you use publicly available code/text, you need to cite the source in your code and report!)

Please submit your solution through Blackboard (Sec 01) or Bottlenose (Sec 02) by the due date shown online. For late submissions you will lose one percentage point per hour after the deadline. This HW is worth 100 points and accounts for 15% of your overall homework score. To encourage early work, you will receive a 10-point bonus if you submit your solution on or before the early submission deadline stated on Blackboard. (Notice that your total score cannot exceed 100 points, but the extra points would compensate for any deductions.) Always package all your solution files, including the report, into a single standard **ZIP** file. Make sure your report is a **PDF** file.

For each program you submit, include complete source code. Do not include input data, output files over 1 MB, or any sort of binaries such as JAR or class files. You must submit a Makefile. The Makefile must be easy to configure to fully build and execute local versions of your code (i.e., AWS is optional). It is recommended that your Makefile call out to another build tool, e.g., Gradle, Maven, or SBT. Comment your Makefile and optionally include a README so TAs can easily configure it to test your program.

PageRank in Spark

We solve the same problem from Assignment 3, but this time in Spark. Your Spark program needs to be written in **Scala**, but can call the input parser (written in Java) from Assignment 3. Overall, the Spark program should perform the following steps:

1. Read the bz2-compressed input.
2. Call the input parser from Assignment 3 on each line of this input to create the graph.
3. Run 10 iterations of PageRank on the graph. The PageRank algorithm has to be written in Scala. As before, it has to deal with dangling nodes.
4. Output the top-100 pages with the highest PageRank and their PageRank values, in decreasing order of PageRank.

When designing your Spark program, think carefully about the RDDs. Important design decisions include if an RDD should be a pairwise RDD, if it should be persisted, if a specific partitioning is needed/helpful, and how to best separate data that does not change from data that does. However, you do *not* need to explore balanced min cut or similar algorithms that attempt to find a graph partitioning that minimizes the number of edges between the partitions in order to minimize network traffic during PageRank computation iterations.

Report

Write a brief report about your findings, using the following structure.

Header

This should provide information like class number, HW number, and your name.

Design Discussion (20 points total)

Describe the steps taken by Spark to execute your source code. In particular, for each method invocation of your Scala Spark program, give a brief high-level description of how Spark applies it to the data. (10 points)

Compare the Hadoop MapReduce and Spark implementations of PageRank. (10 points)

- For each line of your Scala Spark program, describe where and how the respective functionality is implemented in your Hadoop jobs.
- Discuss the advantages and shortcomings of the different approaches. This could include, but is not limited to, expressiveness and flexibility of API, applicability to PageRank, available optimizations, memory and disk data footprint, and source code verbosity.

Performance Comparison (12 points total)

Run your program in Elastic MapReduce (EMR) on the four provided bz2 files, which comprise the full English Wikipedia data set from 2006, using the following two configurations:

- 6 m4.large machines (1 master and 5 workers)
- 11 m4.large machines (1 master and 10 workers)

Report for both configurations the Spark execution time. For comparison, also include the total execution time (from pre-processing to top-k) of the corresponding Hadoop executions from Assignment 3. (4 points)

Discuss which system is faster and briefly explain what could be the main reason for this performance difference. (4 points)

Report the top-100 Wikipedia pages with the highest PageRanks, along with their PageRank values, sorted from highest to lowest, for both the simple and full datasets, from both the Spark and MapReduce execution. Are the results the same? If not, try to find possible explanations. (4 points)

Deliverables

1. The report as discussed above. (1 PDF file)
2. The source code of your Spark program, including an easily-configurable Makefile that builds your programs for local execution. **Make sure your code is clean and well-documented. Messy and hard-to-read code will result in point loss.** In addition to correctness, efficiency is also a criterion for the code grade. (60 points)

3. The syslog files for a successful EMR run for both system configurations. (5 points)
4. *Final* output files from EMR execution only, i.e., only the top-100 pages and their PageRank values. (3 points)