Homework 6: Spark Programming

Introduction to Big Data Systems course

Due: November 6, 2022 23:59 China time. Late submission results in lower (or even no) scores.

For questions or concerns, contact TA (Huanqi Cao, Mingzhe Zhang) by WeChat. Or send an email to caohq18@mails.tsinghua.edu.cn or zmz21@mails.tsinghua.edu.cn if you could not use WeChat.

Overview

In this assignment, you will implement Word Count and Page Rank algorithms with Spark.

You will need to write program in **Python**. You may choose another language, but you should provide your own running script.

Tasks

We have placed the starter code directory at `/data/hw6_src` on the server. Copy it to your home by `cp -r /data/hw6_src ~`.

Task 1: Word Count (30%)

The typical "Hello, world!" app for Spark applications is known as word count. The map/reduce model is particularly well suited to applications like counting words in a document.

All operations in Spark operate on data structures called RDDs, Resilient Distributed Datasets. An RDD is nothing more than a collection of objects. If you read a file into an RDD, each line will become an object (a string, actually) in the collection that is the RDD. If you ask Spark to count the number of elements in the RDD, it will tell you how many lines are in the file. If an RDD contains only two-element tuples, the RDD is known as a "pair RDD" and offers some additional functionality. The first element of each tuple is treated as a key, and the second element as a value. Note that all RDDs are immutable, and any operations that would mutate an RDD will instead create a new RDD.

We have provided starter code in `word_count.py` that loads the input file into a RDD. You are responsible for writing the rest of the application. Your application must return a list of the 10 most frequently occurring words, sorted in descending order of count.

You can use `re.split()` function with the regex `[^\w]+` to split the input text into words.

We have provided a dataset called `pg100.txt` to experiment with

Here is an example of how to run the code:

```
spark-submit word_count.py /data/hw6_data/pg100.txt
```

For `pg100.txt`, the correct output is,

```
('', 198753)
('the', 23288)
('I', 22225)
('and', 18653)
('to', 16373)
('of', 15725)
('a', 12796)
('you', 12186)
('my', 10839)
('in', 10016)
```

Task 2: PageRank (70%)

In this problem, you will learn how to implement the PageRank algorithm in Spark. You can start experimenting with small randomly generated graphs (assume graph has no dead-ends), provided at `/data/hw6_data/small.txt` and `/data/hw6_data/full.txt`. There are 100 nodes (`n = 100`) in the small graph and 1000 nodes (`n = 1000`) in the full graph, and `m = 8192` edges, 1000 of which form a directed cycle (through all the nodes) which ensures that the graph is connected. It is easy to see that the existence of such a cycle ensures that there are no dead ends in the graph. There may be multiple directed edges between a pair of nodes, and your solution should treat them as the same edge. The first column in `/data/hw6_data/full.txt` refers to the source node, and the second column refers to the destination node.

Run the aforementioned iterative process in Spark for 100 iterations (assuming `d = 0.8`). Compute the top 5 node IDs with the highest PageRank scores.

Here is an example of how to run the code:

```
spark-submit page_rank.py /data/hw6_data/small.txt
spark-submit page_rank.py /data/hw6_data/full.txt
```

For a sanity check, we have provided a smaller dataset (`small.txt`). In that dataset, the top node has ID 53 with value approximately 0.0357312 after 100 iterations (you can use this value to help debug).

We will be grading you on your results for full.txt. We give you a file `page_rank.py` to write your code in, with basic starter code that starts your Spark context and reads in the input text file as an RDD. You will also be reporting the total time it took your program to run. The starter code already wraps the code you will write with timing code (report this number in seconds).

Hand-in

Please submit your assignment containing your PDF report and code, and optionally a running script (if you choose to code in another language). Pack everything in a ZIP file.

Points in this homework will solely be based on the correctness of your implementation on the provided input files, and the quality of your writeup,

- 20 points: correctness of Word Count
- 10 points: writeup for Word Count

- 50 points: correctness of PageRank
- 20 points: writeup for PageRank

Please describe your solution in detail in your report. Besides, please tell me how to run your program successfully.