Univ.-Prof. Dr.-Ing. Matthias Boehm
Graz University of Technology
Computer Science and Biomedical Engineering
Institute of Interactive Systems and Data Science
BMVIT endowed chair for Data Management

4 Data Management SS20: Exercise 04 – Large-Scale Data Analysis

Published: May 24, 2020 (last update: May 29, made Q13 less ambiguous)
Deadline: June 16, 2020, 11.59pm

This exercise on large-scale data analysis aims to provide practical experience with distributed data management and large-scale data analysis on top of Apache Spark. The expected result is a zip archive named `DBExercise04_<student_ID>.zip`, submitted in TeachCenter.

4.1 Apache Spark Setup (3/25 points)

As a preparation step, setup Apache Spark and necessary Hadoop client APIs inside an IDE (integrated development environment) of your language choice. This exercise can be done with the Spark language bindings Java, Scala, or Python. For example in Java, you could simply include the maven dependencies `spark-core` and `spark-sql` into your project. On Windows, please download `winutils.exe` from `https://github.com/steveloughran/winutils/tree/master/hadoop-2.7.1/bin`, put it into a directory `<some-path>/hadoop/bin`, and create a new environment variable `HADOOP_HOME=<some-path>/hadoop`. The input data for this exercise is available at `https://mboehm7.github.io/teaching/ss20_dbs/data.zip` (from Exercise 3, based on the schema from Exercise 2).

Partial Results: N/A.

4.2 SQL Query Processing (4/25 points)

In order to further practice basic SQL query processing, please create the following two SQL queries. You get 2 points per query as this is primarily a repetition but note that these queries are the input for tasks 4.3 and 4.4.

- **Q12:** What are the top-5, unique co-author pairs by number of jointly published papers? (return pairs of author names and paper count; ordered descending by paper count)
- **Q13:** Which persons published more than 20 papers at SIGMOD conferences or PVLDB journals between 2014 and 2020? (return names and paper count; ordered descending by paper count)

Partial Results: SQL script `Queries.sql`.

4.3 Query Processing via Spark RDDs (12/25 points)

Spark's basic abstraction for distributed collections are so-called Resilient Distributed Datasets (RDDs). In this task, you should implement the queries Q12 and Q13 from task 4.2 via RDD operations, collect the results in the driver and print the result list to stdout. Please implement these queries as two self-contained functions/methods `executeQ12RDD()` and `executeQ13RDD()` that internally create a `SparkContext` sc, read the files via `sc.textFile()`, and use only RDD[1] operations to compute the query results.

Partial Results: Source file `QueriesRDD.*`.

4.4 Query Processing via Spark SQL (6/25 points)

Spark also provides the high-level APIs `Dataframe` and `Dataset` for SQL processing. In this task, you should implement queries Q12 and Q13 from task 4.2 via Dataset operations, and write the outputs to Parquet files (columnar format) `out12.parquet` and `out13.parquet`. Please implement these queries as two self-contained functions/methods `executeQ12Dataset()` and `executeQ13Dataset()` that internally create a `SparkSession` sc, read the inputs files via `sc.read().format("csv")`, and use only SQL or Dataset operations to compute and write the query results. You might either (1) register the individual input Datasets as temporary views and compute the results directly via SQL, or (2) alternatively use the functional API of Datasets. Both specifications share a common query optimization and processing pipeline.

Partial Results: Source file `QueriesDataset.*`.

4.5 Extra Credit (5 points)

Given the AuthPapers relation in form of two alternative graph representations (provided as `AuthPapersCOO.csv`, and `AuthPapersCSR.csv` in `https://mboehm7.github.io/teaching/ss20_dbs/data2.zip`), write a program to compute the *connected components* of the co-author graph. Your program should leverage Spark, but the API selection is up to you (e.g., RDD operations, SQL, or higher-level libraries like Spark GraphX). The expected output is a text file mapping vertices (author IDs) to components, as well as a summary of the number of components printed to stdout.

Partial Results: Source file `Components.*`.

[1] `https://spark.apache.org/docs/latest/rdd-programming-guide.html`