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

4 Data Management SS22: Exercise 04 – Large-Scale Data Analysis

Published: May 28, 2022 (updates: N/A)
Deadline: June 21, 2022, 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. The entire exercise is *extra credit* for the course data management, and not part of the course databases.

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 include the maven dependencies `spark-core` and `spark-sql`. On Windows, please download `winutils.exe` and other files from `https://github.com/cdarlint/winutils/tree/master/hadoop-3.2.2/bin`, put them into a directory `<some-path>/hadoop/bin`, and create an environment variable `HADOOP_HOME= <some-path>/hadoop`. The schema and input data for this exercise can be reused from Exercise 2: `https://mboehm7.github.io/teaching/ss22_dbs/DataExport.zip`.

Partial Results: N/A (every submission receives these points).

4.2 Query Processing via Spark RDDs (11/25 points)

Apache Spark's basic abstraction for distributed collections are so-called Resilient Distributed Datasets (RDDs). In this task, you should implement the queries **Q02** and **Q05** from Task 2.3 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 `executeQ02RDD()` and `executeQ05RDD()` 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.*`.

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

4.3 Query Processing via Spark SQL (5/25 points)

Spark also provides the high-level APIs `Dataframe` and `Dataset` for SQL processing. In this task, you should implement queries **Q02** and **Q05** from Task 2.3 via Dataset operations, and write the outputs to JSON files `out02.json` and `out05.json`. Please implement these queries as two self-contained functions/methods `executeQ02Dataset()` and `executeQ05Dataset()` that internally create a `SparkSession` sc, read the inputs files via `sc.read().format("csv")`, and uses 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.*`.

4.4 Population Count Prediction (6/25 points)

Given the table PopByCitizenship, create a regression model for predicting the future population for any (district, country, date) combination. In detail, create a distributed machine learning pipeline utilizing Spark MLlib (RDD) or spark.ml (Datasets). This pipeline should split the data into training (2006-2019) and test (2020-2022) sets; apply one-hot encoding to the district and country keys, encode the date as time since `2006-01-01`; and train a linear regression or tree-based regression model using the population counts as targets $\mathbf{y}$. Finally, evaluate your trained model by computing predictions $\hat{\mathbf{y}}$ for the test set, and summarizing the average residuals $\frac{1}{N}\sum(\mathbf{y}-\hat{\mathbf{y}})$, sum of squared residuals $\sum(\mathbf{y}-\hat{\mathbf{y}})^2$, and R^2 (coefficient of determination).

Partial Results: Source file `MLPipeline.*`.