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 Database Systems SS19: Exercise 04 – Large-Scale Data Analysis

Published: June 03, 2019
Deadline: June 25, 2019, 11.59pm CET

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 `DB_Exercise04_<student_ID>.zip`, submitted in TeachCenter.

4.1 Apache Spark Setup (4/25 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/ss19_dbs/data.zip` (from Exercise 3, based on the schema from Exercise 2).

Partial Results: N/A.

4.2 SQL Query Processing (5/25 points)

In order to further practice basic SQL query processing, please create the following two SQL queries. You get 2.5 points per query as this is primarily a repetition but note that these queries are the input for tasks 4.3 and 4.4.

- **Q11:** Compute the list of distinct clubs, the players of team Germany 2014 were affiliated with at the time of the tournament, and return a list of (Club Name, Number of Players), sorted in descending order of the number of players.
- **Q12:** Compute for each world cup tournament its length in terms of the number of days between first and last match, and return a list of (TYear, Host Name, Length) sorted by (Length, Year) in ascending order. Tournaments with multiple hosts should be represented as multiple tuples.

Partial Results: SQL script `Queries.sql`.

4.3 Query Processing via Spark RDDs (10/25 points)

Spark's fundamental abstraction for distributed collections are so-called Resilient Distributed Datasets (RDDs). In this task, you should implement queries Q11 and Q12 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 `executeQ11RDD()` and `executeQ12RDD()` that internally create a `SparkContext` sc, read the files via `sc.textFile()`, and use only RDD 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 Q11 and Q12 from task 4.2 via Dataset operations, and write the outputs to JSON files `out11.json` and `out12.json`. Please implement these queries as two self-contained functions/methods `executeQ11Dataset()` and `executeQ12Dataset()` 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.*`.