ACADEMIC YEAR 2022-2023
SPRING SEMESTER
M111
Big Data Management
Final Project

Deadline:30/6/2023

For the Final project you're going to use Apache spark to execute queries on datasets. There are 2 basic APIs for query execution.

1. RDD API: https://spark.apache.org/docs/2.4.4/rdd-programming-guide.html
2. DataFrame API / SQL API: https://spark.apache.org/docs/2.4.4/sql-programming-guide.html

You can read more about SQL query execution in spark here:

https://spark.apache.org/docs/2.4.4/sqlprogramming-guide.html#running-sql-queries-programmatically

To download the Dataset for the final project you need to execute the following commands in your okeanos master machine:

```
wget https://www.dropbox.com/s/c10t67glk60wpha/datasets2023.tar.gz?dl=0
mv datasets2023.tar.gz?dl=0 datasets.tar.gz
tar -xzf datasets.tar.gz
```

This zip file contains 5 files: movies.csv, movie_genres.csv, ratings.csv, departmentsR.csv, employeesR.csv

The files are in the csv format, executing queries in this format is not efficient. To optimize the data access, traditionally databases load data into special designed binary formats. Spark has a similar approach too; we can convert datasets to a special format named "parquet".

The parquet file formats have major benefits:

1. has a smaller footprint in memory and disk and therefore optimizes I / O (input / output) reducing execution time
2. Maintains additional information, such as statistics on the dataset, which helps on more efficient processing

The parquet file format is a columnar file format, you can read more about it here https://parquet.apache.org/ . On how to read and write parquet files you can find information here: https://spark.apache.org/docs/2.4.4/sql-data-sources-parquet.html

Explaining the dataset

This is a brief explanation of what the datasets looks like

For the file movies.csv:

This file contains the movies data. This is the first entry of the movies.csv file:

862	Toy Story	Led by W	1995	81	30000000	373554033	21.94694

First is the movies id, then the name of the movie, the description, the release year of the movie, the duration of the movie, how much the movie costed to produce, what the revenue of the movie was and finally the popularity of the movie.

For the file ratings.csv:

This file contains the users' ratings for the movies. This is the first entry of the ratings.csv file:

1	110	1	1425941529

First is the user's id, then the movie id that the rating is for, then the rating value and finally the timestamp that the rating was created.

For the file movie_genres.csv:

This file contains the movies genres. This is the first entry of the movie_genres.csv:

862	Animation

First is the movie's id and then the genre of the movie.

Part 1

Task 1:

Create a files directory on HDFS and then upload the CSVs to the files directory: provide the commands needed for the directory creation and the files upload. Provide a print screen that shows the csv files in the directory you created (**0.5 points**). Convert the files to parquet and upload them on HDFS (**0.5 points**)

Task 2:

Using RDDs write code to answer the following queries (Q1-Q5). You can use the csv or parquet files you uploaded on HDFS (**2.75 points, 0.55x5**)

Task 3:

Using DataFrames write code to answer the following queries (Q1-Q5) using the parquet files you created and uploaded to HDFS (**2.75 points, 0.55x5**)

Task 4:

For every query (Q1-Q5) measure the execution time of each scenario:

1. Map/Reduce - RDD API
2. Spark SQL on csv files (you need to define the schema of each file)
3. Spark SQL on parquet files

Create a bar chart with the execution times grouped by query number and write a paragraph explaining the results (**0.5 points**).

Q1: For every year after 1995 print the difference between the money spent to create the movie and the revenue of the movie (revenue – production cost).

INFO: For this query you will need to filter out the entries that have zero values for the columns revenue and/or cost

Q2: For the movie "Cesare deve morire" find and print the movies id and then search how many users rated the movie and what the average rating was?

Q3: What was the best in terms of revenue Animation movie of 1995?

INFO: For this query you will need to filter out the entries that have zero values for the columns revenue and/or cost

Q4: Find and print the most popular Comedy movies for every year after 1995.

INFO: For this query you will need to filter out the entries that have zero values for the columns revenue and/or cost

Q5: For every year print the average movie revenue

INFO: For this query you will need to filter out the entries that have zero values for the columns year and revenue

Part 2

In this part of the final project, you will evaluate different join algorithms.

Task 1:

Write Map/Reduce jobs using the RDD API that implement the broadcast join (**1 point**) and the repartition join (**1 point**) of datasets employessR.csv and departmentsR.csv. You can read more about the algorithms and their implementations here (for broadcast join Paragraph 3.2 , Pseudocode A.4. and for repartition join Paragraph 3.1, Pseudocode A.1)

Datasets explanation:

For the datasets employessR.csv and departmentsR.csv:

First line of employeesR:

1	Elizabeth Jordan	7

Employees id, name and department id

First line of departmentsR.csv:

1	Dep A

Department ID, Department Name

INFO: Join the two datasets on the department id key

Task 2:

Spark SQL includes different join algorithms implementations. It also includes a query optimizer named "Catalyst". With the help of catalyst Spark SQL chooses the best join algorithm based on the tables we want to join. We can stop Spark from choosing a join algorithm automatically.

You can use the following script to stop spark from choosing a join algorithm automatically. Execute the following query with the join optimization enabled and disabled. Create a bar graph with the execution time and explain the results. Take a screenshot of the SQL plan printed by the command spark.sql(query).explain() and give a brief explanation of the SQL plan (**1 point**).

INFO: you need to change every variable is defined with <> and provide your own custom values

To run the query with join optimization on: spark-submit <Name_of_the_file>.py Y

To run the query with join optimization off: spark-submit <Name_of_the_file>.py N

```
from pyspark.sql import SparkSession
import sys, time
disabled = sys.argv[1]
spark = SparkSession.builder.appName('query1-sql').getOrCreate()
if disabled == "Y":
    spark.conf.set(<conf_name>,<conf_value>)
elif disabled == 'N':
    pass
else:
    raise Exception ("This setting is not available.")

df = spark.read.format("parquet")
df1 = df.load(<ratings_parquet_file_on_hdfs>)
df2 = df.load(<movie_genres_file_on_hdfs >)
df1.registerTempTable("ratings")
df2.registerTempTable("movie_genres")
sqlString = \
"SELECT * " + \
"FROM " + \
" (SELECT * FROM movie_genres LIMIT 100) as g, " + \
" ratings as r " + \
"WHERE " + \
" r._c1 = g._c0"
t1 = time.time()
spark.sql(sqlString).show()
t2 = time.time()
spark.sql(sqlString).explain()
print("Time with choosing join type %s is %.4f sec."%("enabled" if
disabled == 'N' else "disabled", t2-t1))
```

1. The deadline of the final project is: 30/6/2023
2. This assignment constitutes 50% of the total grade of the course.
3. There may be an oral examination of this assignment, if needed.
4. You can ask questions about the assignment on the eclass forum "Project 2023" https://eclass.uoa.gr/modules/forum/viewforum.php?course=DI508&forum=65207 so that any question/answer is available to everyone.
5. Submit a zip containing (a) a folder with the developed code named "src" (you can create separate code files for every task-part of the assignment e.g. Q1RDD.py, Q1DF.py …), (b) a folder named "output" that will contain the execution outputs of all the parts of the assignment and (c) a PDF named "Report" that will contain screenshots and explanations for each part of the assignment.