Big Data Platforms - Home Assignment 2

Due to: 09.12.2021

Homework Tasks

Initial steps

1. Write Python code to create 20 different CSV files, **myCSV[Number].csv** where each file contains 10 records. The schema is ('firstname','secondname',city') and values should be randomly chosen from the lists
   firstname : [John, Dana, Scott, Marc, Steven, Michael, Albert, Johanna]
   city: [New York, Haifa, München, London, Palo Alto, Tel Aviv, Kiel, Hamburg]
   secondname: any value
2. Create **mapreducetemp** and **mapreducefinal** folders in your laptop

Task 1: MapReduceEngine

1. Write Python code to create an SQLite database with the following table

   ```
   Table: temp_results
   schema: (key:TEXT,value:TEXT)
   ```

2. Create a Python class **MapReduceEngine** with method

   ```
   def execute(input_data, map_function, reduce_function, params)
   ```

   such that
   - **input_data** is an array of elements
   - **map_function** is a pointer to the Python function that returns a list where each entry of the form **(key,value)**
   - **reduce_function** is pointer to the Python function that returns a list where each entry of the form **(key,value)**
   - **params** are parameters to the map_function of the form params = {key:value}

3. Implement the following functionality in the **execute(..)** function

   a) For each **key** from the **input_data**, start a new Python thread that executes **map_function(key)**
   b) Each thread will store results of the **map_function** into **mapreducetemp/part-tmp-X.csv** where X is a unique number per each thread.
   c) Keep the list of all threads and check whether they are completed.
   d) Once all threads completed, load content of all CSV files into the **temp_results** table in SQLite.

Remark: The easiest way is to loop over all CSV files and load them into Pandas first, then load into SQLite, example:

data = pd.read_csv(path to csv)
data.to_sql('temp_results',sql_conn, if_exists='append',index=False

e) Write SQL statement that generates a sorted list by key of the form (key, value) where value is concatenation of ALL values in the **value** column that match specific key
For example, if table has records

John	myCSV1.csv
Dana	myCSV5.csv
John	myCSV7.csv

Then SQL statement will return *('John','myCSV1.csv, myCSV7.csv')*
Remark: use GROUP_CONCAT and also GROUP BYORDER BY

f) Start a new thread for each value from the generated list in the previous step, to execute **reduce_function(key,value)**
g) Each thread will store results of **reduce_function** into **mapreducefinal/part-X-final.csv** file
h) Keep list of all threads and check whether they are completed.
i) Once all threads completed, print on the screen "MapReduce Completed"

Task 2: Implement the MapReduce Inverted index of the JSON documents

1. Write a function **inverted_map(document_name, column_index)** which reads the CSV document from the local disc and return a list that contains entries of the form **(key_value, document name)** for the specific column_index provided.
For example, if column_index = 1 and myCSV11.csv document has values like

firstname	secondname	city
Michael	Vernik	Tel Aviv
Johanna	Vernik	Hamburg
Marc	Friedman	New York
Steven	Friedman	Palo Alto
Michael	Friedman	MvºnchenMvºnchen
Michael	Vernik	London
Dana	Friedman	Tel Aviv
Steven	Vernik	Haifa
Marc	Vernik	Kiel

{Then **inverted_map('myCSV11.csv', column_index=1)** function will return a list

of the form

key	value			
Michael	/Users/gilv/Dev/DataStore/csv/myCSV11.csv			
Johanna	/Users/gilv/Dev/DataStore/csv/myCSV11.csv			
Marc	/Users/gilv/Dev/DataStore/csv/myCSV11.csv			
Steven	/Users/gilv/Dev/DataStore/csv/myCSV11.csv			
Michael	/Users/gilv/Dev/DataStore/csv/myCSV11.csv			
Michael	/Users/gilv/Dev/DataStore/csv/myCSV11.csv			
Dana	/Users/gilv/Dev/DataStore/csv/myCSV11.csv			
Steven	/Users/gilv/Dev/DataStore/csv/myCSV11.csv			
Marc	/Users/gilv/Dev/DataStore/csv/myCSV11.csv			

2. Write a reduce function **inverted_reduce(key, documents)**, where the field "documents" contains a list of all CSV documents per given key. This list might have duplicates. Reduce function will return new list without duplicates.

Task 3: Submit your first MapReduce-

1. Create Python list *input_data: 'myCSV0.csv',.. ,'myCSV19.csv']*
2. Submit MapReduce as follows

```
mapreduce = MapReduceEngine()
status = mapreduce.execute(input_data, inverted_map, inverted_index, params =
{'column':1})

print(status)
```

3. "MapReduce Completed" should be printed and **mapreducefinal** folder should contain the result files.
4. Delete all temporary data from **mapreducetemp** folder and delete SQLite database

Task 4:

The phase where MapReduceEngine reads all temporary files generated by maps and sort them to provide each reducer a specific key is called the shuffle step. What would be the main problem of MapReduce when processing Big Data, if there is no shuffle step at all, meaning reducers will directly read responses from the mappers.

Good luck!