Assignment 3: Distributed Batch Processing Using Apache Spark

CS328 - Distributed and Cloud Computing

Deadline: 23:59, Dec 15, 2024

1 Requirements

You are provided with the dataset (`parking_data_sz.zip`) that contains information about parking lot utilization in Shenzhen. The dataset is in `.csv` format, with the entries (lines) storing information about the parking location and parking duration of a vehicle. The following table describes the data stored for each vehicle (columns):

Name	**Description**	**Example**
`in_time`	Time vehicle entered the parking lot	`"2018-09-01 10:10:00"`
`out_time`	Time vehicle left the parking lot	`"2018-09-01 12:00:00"`
`berthage`	Unique ID of the lot used by the vehicle	`"201091"`
`section`	Section of the lot used by the vehicle	" 荔园路 (蛇口西段)"
`admin_region`	District of the lot used by the vehicle	" 南山区"

You are required to perform five data analysis tasks on this dataset using *Apache Spark's* python API *PySpark* and outputting the results in five separate .csv files. Specifically:

1. Output the total number of *berthages* for each *section*. The output file should have the columns **`section`** and **`count`**. (10 points)

2. Output all unique *berthages* for each *section*. The output file should have the columns **`berthage`** and **`section`**. (20 points)

3. Output the average parking time for each *section*. The output file should have the columns **`section`** and **`avg_parking_time`**. The average parking time should be an integer representing seconds (i.e., cast the value to Python `int`). (20 points)

4. Output the average parking time for each *berthage*, sorted in descending order. The output file should have the columns **`berthage`** and **`avg_parking_time`**. The average parking time should be an integer representing seconds (i.e., cast the value to Python `int`). (20 points)

5. For each *section*, output the total number of *berthages* in-use ("in use" means there is at least one vehicle parked in that *berthage*) and the percentage of in-use *berthages* in that *section*, in a one-hour interval (e.g. during 09:00:00-10:00:00). The output file should have the columns `start_time`, `end_time`, `section`, `count` and `percentage`. The percentage value should be rounded to one decimal place (e.g. 67.8%). The data format of `start_time` and `end_time` should match the one in the original dataset "`YYYY-MM-DD HH:MM:SS`" (20 points)

   - **Sub-task:** Select **three** sections from the results and plot (one or three figures), with time as the X axis and percentage in use as the Y axis. Analyse the results and state any interesting trends that you identify in the report. (10 points)

For each task, briefly explain the implementation logic and the *PySpark* calls you used to achieve your result. Include screenshots for each of your Spark job's DAG, using the Spark Web UI (example).

For the above, you can use the Standalone version of PySpark in combination with your local file system.

1.1 Bonus (10 points)

Deploy a Hadoop HDFS cluster with YARN and Spark in Docker. Store the dataset in HDFS and submit the Spark scripts you wrote for the above tasks to the cluster using YARN. Include the following in your report:

- an explanation of the process you used to set up the cluster
- screenshots of the containers running and of the YARN-UI showing the successful execution of your Spark jobs
- an explanation of how the execution of Spark jobs submitted to YARN differs from jobs executed in the Standalone version.

1.2 Implementation notes

- This assignment does not impose any requirement on the data representation used (RDD or DataFrames); you can choose whichever is more convenient for the task.
- Both .py scripts and Python notebooks are allowed. Use whichever works best for you.
- If you are using .py scripts, you can add a `time.sleep(X)` call at the end of the script to keep Spark running for enough time to capture the screenshots of the DAG.
- If you are using actions to print intermediate RRD's or DataFrames (something that you should do) make sure to remove all of them (keeping only the final action)

before taking a screenshot of the DAG in SparkUI. Otherwise, your processing DAG will be split into smaller DAGs (one for every action).

2 Limitations

- Using pure Python and Python libraries besides PySpark for data processing is prohibited. **ALL** data processing **MUST** be done using the PySpark API and nothing else. Pure Python should **ONLY** be used to write the `.csv` files with the results.
- There is no 'hard' limitation on how many transformations you can use to complete the tasks. However, you should aim for the simplest solution. Highly convoluted solutions that could be easily avoided by using a transformation already provided by Spark can result in a loss of marks.

3 Hints

1. Filter out invalid data (e.g., `out_time` $\leq$ `in_time`) in advance.
2. Consider if converting the data (`in_time`, `out_time`) to (`in_time`, `parking_time_length`) can simplify some tasks.
3. Visit:
   - https://spark.apache.org/docs/latest/api/python/reference/pyspark.html
   - https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/index.html

   to find useful PySpark RRD and DataFrame transformations and actions.

4 What to Submit

1. Source code: `.py` scipt(s) OR `.ipynb` notebook(s).
2. A report in **PDF** format.
3. Five separate `.csv` files containing the results of the five requirements. The names of the files should be: `r1.csv, r2.csv, r3.csv, r4.csv, r5.csv`.
4. **BONUS TASK**: If you have completed the bonus task, also include all relevant `Dockerfile` and `docker-compose.yaml` files.

Pack all files into `SID_NAME_A3.zip`, where `SID` is your student ID and `NAME` is your name (e.g., 11710106_ 张三 _A3.zip).