[Project 1] [Phase 2]

CSE 511: Data Processing at Scale - Spring 2023

Available: 02/19/2023 **Due Date**: 03/05/2023 11:59 PM **Points: 100**

Introduction & Background

In the first phase of this project, you learned to run multiple spatial queries on the large geo-database of the taxi firm that hired you. Now, the firm has contacted you again to extend the solution. Your team now needs to perform further analysis for them by locating passenger hot spots. There is **also a Bonus opportunity of 20% grade points** in this phase of the project as well.

Problem Description

In this phase of the project, you are required to do spatial **hot spot analysis**. In particular, you need to complete two different hot spot analysis tasks.

- **Hot zone analysis**: This task will need to perform a range join operation on a rectangle datasets and a point dataset. For each rectangle, the number of points located within the rectangle will be obtained. The hotter rectangle means that it includes more points. So this task is to calculate the hotness of all the rectangles.
- **Hot cell analysis** : This task will focus on applying spatial statistics to spatio-temporal big data in order to identify statistically significant spatial hot spots using Apache Spark.

The topic of this project phase is inspired from ACM SIGSPATIAL GISCUP 2016. Though we have special requirements in this assignments which are different from the GIS Cup.
Please find the following references on the problem description:

- A detailed problem definition is available here
- Evaluation Instructions are available here

Problem Statement

As stated in the Problem Definition page, in this task, you are asked to implement a Spark program to calculate the Getis-Ord statistic of NYC Taxi Trip datasets. We call it "Hot cell analysis". To reduce the computation power need, we made the following changes:

- The input will be a monthly taxi trip dataset from 2009 - 2012. For example, `yellow_tripdata_2009-01_point.csv`, `yellow_tripdata_2010-02_point.csv`
- Each cell unit size is 0.01 * 0.01 in terms of latitude and longitude degrees.

- You should use 1 day as the Time Step size. The first day of a month is step 1. Every month has 31 days.
- You only need to consider Pick-up Location.
- We don't use Jaccard similarity to check your answer. However, you don't need to worry about how to decide the cell coordinates because the code template generated cell coordinates. You just need to write the rest of the task.

How to work on the assignment and test your code

You are provided with a docker image with all the packages (java, scala, spark) installed and ready to go.

- You will find the template code also in the docker. Path: `/root/cse511`
- The main function/entrace is `cse511.Main` scala file.
- DO NOT DELETE any existing code in the coding template unless you see this "YOU NEED TO CHANGE THIS PART"
- In the code template, for **Hot zone analysis**
    - **You need to change `HotzoneAnalysis.scala` and `HotzoneUtils.scala`.**
    - The template code has loaded the data and wrote the first step, range join query, for you. Please finish the rest of the task.
    - The output DataFrame should be sorted by you according to "rectangle" string.
    - **Input format:** The input point data can be any small subset of NYC taxi dataset.
    - **Output format:** All zones with their count, sorted by "rectangle" string in an ascending order.
- In the code template, for **Hot cell analysis**,
    - **You need to change `HotcellAnalysis.scala` and `HotcellUtils.scala`.**
    - The template code has loaded the data and decided the cell coordinate, x, y, z and their min and max. Please finish the rest of the task.
    - The output DataFrame should be sorted by you according to G-score. The coding template will take the first 50 to output. DO NOT OUTPUT G-score.
    - **Input format:** The input point data is a monthly NYC taxi trip dataset (2009-2012) like "`yellow_tripdata_2009-01_point.csv`"
    - **Output format**: The coordinates of top 50 hotest cells sorted by their G score in a descending order. Note, DO NOT OUTPUT G score.
      -7399,4075,15
      -7399,4075,29
      -7399,4075,22
- An example input and answer are places in "`testcase`" folder in the provided docker template

Note:

- **Point data**: the input point dataset is the pickup point of New York Taxi trip datasets. The data format of this phase is the original format of NYC taxi trip which is different from

Phase 1. Find the data in the .zip file: Yellow_tripdata_2009-01_point.csv (updated to new 2023 canvas link)

- **Zone data** (only for hot zone analysis): at "`src/resources/zone-hotzone`" in the provided docker template.

How to run your code on Apache Spark using "spark-submit"

If you are using the Scala template, note that:

1. You **only have to replace the logic** (currently is "true") in all User-Defined Functions.
2. The main function in this template takes the **dynamic length of parameters and** the number/order of tasks do not matter.
3. Input parameters:
    - Output file path (**Mandatory**): `result/output`
    - Task name: `hotzoneanalysis` or `hotcellanalysis`
    - If the task is `hotzoneanalysis` it requires two parameters
        1. NYC taxi data path
        2. Zone path
    - If the task is `hotcellanalysis`it requires one parameter
        1. NYC taxi data path
4. The number of queries and the order of queries in the input does not matter. The code template will detect the corresponding query and call it!
5. Here is an example that tells you how to submit your jar using "`spark-submit`"

```
spark-submit CSE511-assembly-0.1.0.jar result/output
hotzoneanalysis src/resources/point-hotzone.csv src/resources/zone-hotzone.csv
hotcellanalysis src/resources/yellow_tripdata_2009-01_point.csv
```

How to submit your code to Spark

If you are using the Scala template

1. Go to the project folder in the provided docker image
2. Run `sbt assembly`
3. Find the packaged jar in "`./target/scala-2.11/CSE511-assembly-0.1.0.jar`"
4. Submit the jar to Spark using the Spark command "spark-submit"
5. You must revert Steps 3 and 4 in "How to debug your code in IDE" and recompile your code before using spark-submit!!!

Grading:

- The example input has already been provided to you in the docker image as well. The format for our input will be similar; however test cases, might be different. Make sure your code works in different use cases.

- This is a group project, and contribution from every group member is a must. We will use GitHub to keep track of the contributions of individual team members.
- Each spatial hot spot analysis carries equal weight of 50% of grade points.
- Two example datasets and a test case file and answer are also available in the docker for your reference. `/src/resources` and are also available here

- In case any member has 0 **valid** contributions in the Project-1 repository, the particular individual member will be awarded a 0 in the Project.
- REAME.md file submissions are **NOT** valid contributions. Code comments are **NOT** valid contributions
- You need to make sure your code can compile and package by entering sbt assembly. We will run the compiled package on our cluster directly using "**spark-submit**". If your code cannot compile and package, you will **not** receive any points

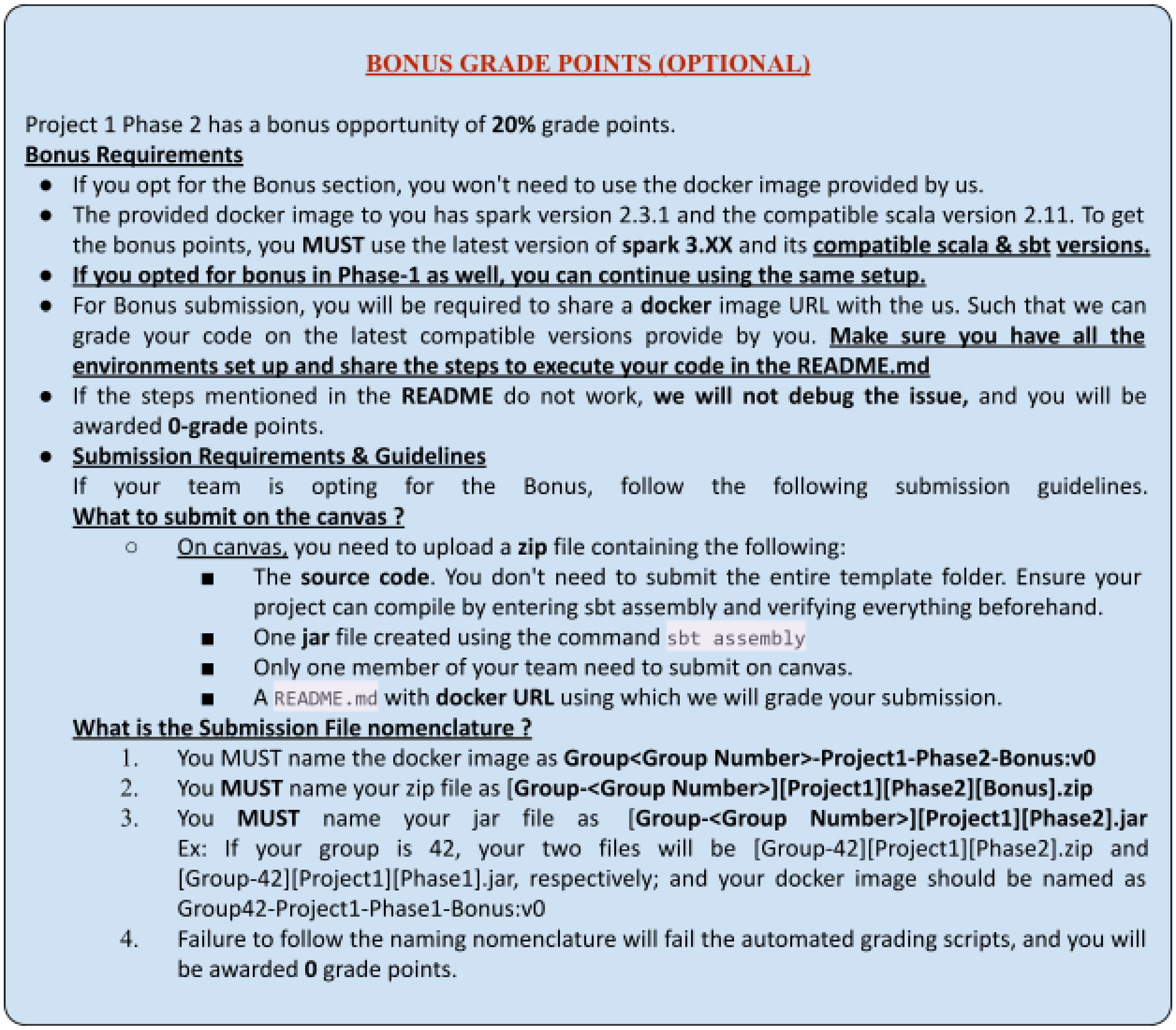
Submission Requirements & Guidelines

1. This is a **group** assignment.
2. **What to maintain on the GitHub private repository ?**
    1. Maintain your code on the provided private GitHub repository in the [SPRING-2023] [CSE511] Data Processing at Scale)
    2. While there is only 1 file you need to update, you can choose to maintain the entire template folder provided in your groups GitHub repository.
    3. Use the same repository as for Project-1 Phase-1, however, make sure to use a new **branch called "phase-2"** for any work done during this phase of the project.
    4. Make sure each group member has made valid contributions to the project and on the GitHub repository.
3. **What to submit on the canvas ?**
    1. On the canvas you need to upload a zip file containtaing the following:
        1. The **source code**. You dont need to submit the entire template folder. Ensure your project can compile by entering sbt assembly and verifying everything beforehand.
        2. One **jar** file created using the command `sbt assembly`
        3. Only one member of your team need to submit on canvas.
4. **What is the Submission File nomenclature ?**
    1. You **MUST** name your zip file as [**Group-<Group Number>][Project1][Phase2].zip**
    2. You **MUST** name your jar file as [**Group-<Group Number>][Project1][Phase2].jar**
    Ex: If your group is 42, your two files will be [Group-42][Project1][Phase2].zip and [Group-42][Project1][Phase2].jar, respectively.
    3. Failure to follow the naming nomenclature will fail the automated grading scripts, and you will be awarded **0** grade points.

Submission Policies

1. Late submissions will ***absolutely not*** be graded (unless you have verifiable proof of emergency). It is much better to submit partial work on time and get partial credit than to submit late for no credit.
2. Every group needs to ***work independently*** on this exercise. We encourage high-level discussions among students to help each other understand the concepts and principles. However, a code-level discussion is prohibited, and plagiarism will directly lead to the failure of this course. We will use anti-plagiarism tools to detect violations of this policy

Bonus!



BONUS GRADE POINTS (OPTIONAL)

Project 1 Phase 2 has a bonus opportunity of **20%** grade points.

Bonus Requirements

- If you opt for the Bonus section, you won't need to use the docker image provided by us.
- The provided docker image to you has spark version 2.3.1 and the compatible scala version 2.11. To get the bonus points, you **MUST** use the latest version of **spark 3.XX** and its **compatible scala & sbt versions.**
- **If you opted for bonus in Phase-1 as well, you can continue using the same setup.**
- For Bonus submission, you will be required to share a **docker** image URL with the us. Such that we can grade your code on the latest compatible versions provide by you. **Make sure you have all the environments set up and share the steps to execute your code in the README.md**
- If the steps mentioned in the **README** do not work, **we will not debug the issue,** and you will be awarded **0-grade** points.
- **Submission Requirements & Guidelines**
  If your team is opting for the Bonus, follow the following submission guidelines.
  What to submit on the canvas ?
  - On canvas, you need to upload a **zip** file containing the following:
    - The **source code**. You don't need to submit the entire template folder. Ensure your project can compile by entering sbt assembly and verifying everything beforehand.
    - One **jar** file created using the command `sbt assembly`
    - Only one member of your team need to submit on canvas.
    - A `README.md` with **docker URL** using which we will grade your submission.

  What is the Submission File nomenclature ?
  1. You MUST name the docker image as **Group<Group Number>-Project1-Phase2-Bonus:v0**
  2. You **MUST** name your zip file as **[Group-<Group Number>][Project1][Phase2][Bonus].zip**
  3. You **MUST** name your jar file as **[Group-<Group Number>][Project1][Phase2].jar** Ex: If your group is 42, your two files will be [Group-42][Project1][Phase2].zip and [Group-42][Project1][Phase1].jar, respectively; and your docker image should be named as Group42-Project1-Phase1-Bonus:v0
  4. Failure to follow the naming nomenclature will fail the automated grading scripts, and you will be awarded **0** grade points.