FIT3182 – Big Data Management & Processing

Assignment 2 Specifications

Due: Tuesday, 21st May 2024, 9:30 AM (MYT)
Worth: 30% of the overall unit assessment marks

Background

StopFire is a campaign started by Monash University to predict and stop fires in Victorian cities. They have employed sensors in different cities of Victoria and have collected a large amount of data. The data is so big that their techniques have failed to provide the results on time to predict fire. They have hired us as *the data analyst* to migrate their data to the NoSQL database (MongoDB). They want us to analyse their data and provide them with results. In addition, they want us to build an application, a complete setup from streaming to storing and analysing the data for them using Apache Kafka, Apache Spark Streaming, and MongoDB.

What you are provided with

- Five datasets:
  - hotspot_historic.csv
  - climate_historic.csv
  - hotspot_AQUA_streaming.csv
  - hotspot_TERRA_streaming.csv
  - climate_streaming.csv
- These files are available in Moodle in the Assessment section for Assignment 2. You can also access these files here.

Information on Dataset

Climate data is recorded on a daily basis, whereas Fire data is recorded based on the occurrence of a fire on a particular day. Therefore, for one climate data, there can be zero or many fire data. All climate data is an average value for the particular day except for max wind speed.

The data is NOT a row per weather station basis. You can simply think of it as Station 1 was reporting data for X number of days, and then Station 2 started reporting data because Station 1 was shut down for instance. streaming data

Global Horizontal Irradiance (GHI) is the total solar radiation incident on a horizontal surface.

Total precipitation (rain and/or melted snow) reported during the day in inches and hundredths; will usually not end with the midnight observation --i.e., may include the latter part of the previous day.

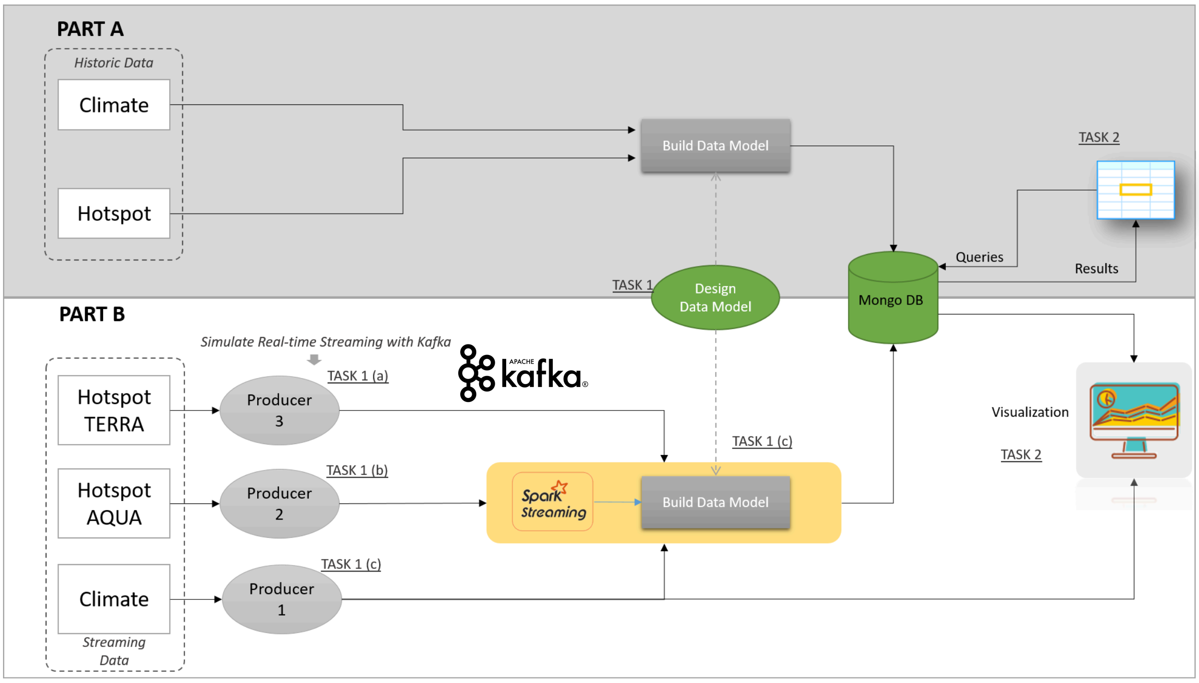
.00 indicates no measurable precipitation (includes a trace). **Missing = 99.99** (*For metric version, units = millimetres to tenths & missing = 999.9.)*

Note: Many stations do not report '0' on days with no precipitation --therefore, '99.99' will often appear on these days. Also, for example, a station may only report a 6-hour amount for the period during which rain fell. See Flag field information below the source of data.

- A = 1 report of 6-hour precipitation amount.
- B = Summation of 2 reports of 6-hour precipitation amount.
- C = Summation of 3 reports of 6-hour precipitation amount.
- **D = Summation of 4 reports of 6-hour precipitation amount.**
- E = 1 report of 12-hour precipitation amount.
- **F = Summation of 2 reports of 12-hour precipitation amount.**
- **G = 1 report of 24-hour precipitation amount.**
- H = Station reported '0' as the amount for the day (eg, from 6-hour reports), but also reported at least one occurrence of precipitation in hourly observations --this could indicate a trace occurred but should be considered as incomplete data for the day.
- I = Station did not report any precipitation data for the day and did not report any occurrences of precipitation in its hourly observations --it's still possible that precipitation occurred but was not reported.

Architecture

The overall architecture of the assignment setup is represented by the following figure.



Part A of the assignment consists of reading data from csv files, building a model, saving to MongoDB and running various queries against the database. **Part B** of the assignment consists of simulating real-time streaming of climate and hotspot data and processing it with Apache Kafka and Spark Streaming. Both approaches should use the same data model and thus the same database.

Part A (20%)

Task 1. MongoDB Data Model (5%)

1. Based on the two data sets provided i.e. *hotspot_historic.csv* and *climate_historic.csv*, design a suitable data model to support the efficient querying of the two data sets in MongoDB. Justify your data model design.
   The output of this task should be
   - *An example of the data model.*
   - *The justification for choosing that data model.*

Task 2. Querying MongoDB using PyMongo (15%)

1. Write a python program to read the data from *hotspot_historic.csv* and *climate_historic.csv* and load them to the new database **(e.g., fit3182_assignment_db)**. The collection(s) in **fit3182_assignment_db** will be based on the document model you have designed in Task A1.
   - Please use a `csv` library to read the files.
   - Please DO NOT use the mongo aggregation query to do this task.

2. Write queries to answer the following tasks on **fit3182_assignment_db** and corresponding collection(s). You need to write the queries as a Python program using the `pymongo` library in Jupyter Notebook.
   a. Find climate data on *12th December 2023*.
   b. Find the *latitude*, *longitude*, *surface temperature* (°C), and *confidence* when the surface temperature (°C) was between *65 °C* and *100 °C*.
   c. Find *date*, *surface temperature (°C)*, *air temperature (°C)*, *relative humidity* and *max wind speed on* 15th and 16th of December 2023.
   d. Find *datetime*, *air temperature (°C)*, *surface temperature (°C)* and *confidence* when the *confidence* is between 80 and 100.
   e. Find the top 10 records with the highest *surface temperature (°C).*
   f. Find the number of fires each day. You are required to only display *the total number of fires* and *the date* in the output.
   g. Find the records of fires where the *confidence* is below 70.
   h. Find the *average surface temperature (°C)* for each day. You are required to only display *average surface temperature (°C)* and *the* date in the output.
   i. Find the top 10 records with the lowest *GHI.*
   j. Find the records with a *24-hour precipitation* recorded between 0.20 to 0.35.
3. The dataset we have is small for now. However, the number of stations and data points will increase significantly once we deploy our database and application in production. Considering those queries in **Task 2** as typical use cases, think about

how to optimise your mongodb database. Based on your data model and typical use cases, add simple or compound indexes to your collection. You also need to consider the high data ingestion rate and design your indexes accordingly. Please justify your index design.

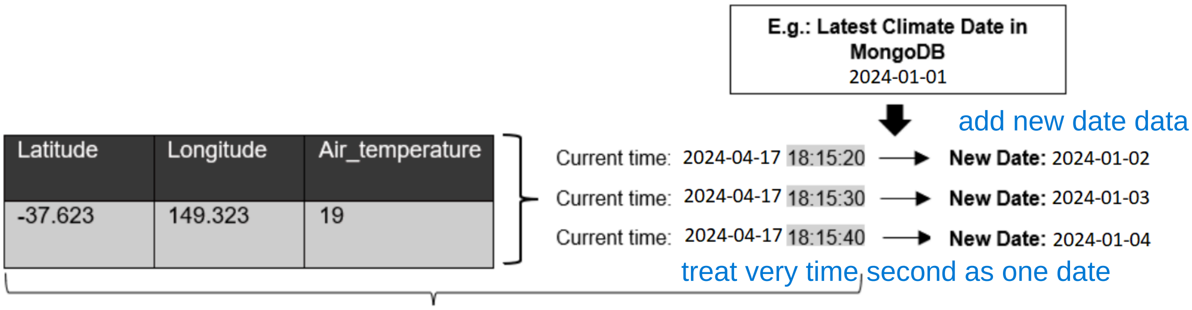
Combine all your answers for Task 1-3 into a single Jupyter Notebook file called **Assignment_PartA.ipynb**.

Part B (60%)

Task 1. Processing Data Stream (45%)

In this task, we will implement multiple Apache Kafka producers to simulate the real-time streaming of the data, which will be processed by the **Apache Spark Structured Streaming** client and then inserted into MongoDB.

Important : *In this task, you are required to use the same data model from Part A. To make the streaming data consistent for the model, you will need to make some changes to the streaming data before building the model or inserting it to mongodb. The figure below gives an example of the date modification.*


E.g.: Latest Climate Date in MongoDB
2024-01-01
add new date data
Latitude
Longitude
Air_temperature
-37.623
149.323
19
Current time: 2024-04-17 18:15:20
New Date: 2024-01-02
Current time: 2024-04-17 18:15:30
New Date: 2024-01-03
Current time: 2024-04-17 18:15:40
New Date: 2024-01-04
treat very time second as one date

Real-time Streaming Data

First, find the **latest date** in the climate data that was inserted into the MongoDB database you created in Part A, Task 2. *Treat every 10 seconds data as **1 day***. Therefore, you will have to create a new date for each 10 seconds data (e.g. new_date = latest date in climate data + 1 day). For the next batch of 10 seconds, we will have a new_date = new_date + 1 day and so on.

Now, you are required to implement three Kafka-producers to simulate the real-time streaming of data.

a. **Event Producer 1:** Write a python program that loads all the data from *climate_streaming.csv* and randomly (with replacement) feed the data to the stream every 10 seconds. You will need to append additional information such as producer information to identify the producer and created date. Save the file as **Assignment_PartB_Producer1.ipynb.**

b. **Event Producer 2:** Write a python program that loads all the data from *hotspot_AQUA_streaming.csv* and randomly (with replacement) feed the data to the stream every *n* seconds, where *n* is some random number of your choosing.

every random time to send next data

AQUA is the satellite from NASA that reports latitude, longitude, confidence and surface temperature of a location. You will need to append additional information such as producer information to identify the producer and a **randomly** created time. Note that you do not have to append any date information as this is inferred from a processed batch as described below, Save the file as **Assignment_PartB_Producer2.ipynb.**

d. week 11?

Merge document

c. **Event Producer 3:** Write a python program that loads all the data from *hotspot_TERRA_streaming.csv* and randomly (with replacement) feeds the data to the stream every *n* seconds, where *n* is some random number of your choosing. TERRA is another satellite from NASA that reports latitude, longitude, confidence and surface temperature of a location. You will need to append additional information such as producer information to identify the producer and a **randomly** created time. Note that you do not have to append any date information as this is inferred from a processed batch as described below, Save the file as **Assignment_PartB_Producer3.ipynb.**

1 and more hotspot and 1 climate -> 1. filter by proximity   0 hotspot -> store climate into mongDB

hotpot has local infor, interest only hotspot near climate, filter proximity by geoHash function return string

precision: 3
3 chars from get gHash
from each element
crop out the sensor
not for that climate
store the climate to MongoDB

every 10 seconds

d. **Streaming Application:** Write a streaming application using the Apache Spark Structured Streaming API which processes data in **daily batches**. Each batch should contain 1 Climate report[1] and 0 or more Hotspot reports (from Producer 2 or 3). The streaming application should process the data as follows:

- Group the Climate and Hotspots streams based on location[2] and create the data model developed in Part A. Assume that all Hotspot reports in a batch window were generated for the current day. Use geohash precision 3 for Climate-to-Hotspot matching. You should drop any Hotspot reports that are not proximately close to the Climate report.
- If there is no Hotspot report in a batch (only Climate report is present), it implies that there was no fire for that day and we can store the Climate report into MongoDB straight away.
- Due to synchronisation issues in the communication between the satellites and the receiving station, the AQUA and TERRA satellites may unknowingly produce multiple reports for the same Hotspot event. This is characterised as Hotspot reports with **similar location** (geohash precision 5) and **created time** (<=10 minutes apart). In such cases, you should merge[3] these reports by averaging the 'surface temperature' and 'confidence'. You are free to choose the method to merge the other fields in the reports (i.e. averaging, first/last[4] etc.).

similar location determined
1. report created time within 10 mins
2. geoHash, the same for
precision 5

merge documents

[1] Intuitively, only a single Climate report should be present in a batch. However, due to the simulated aspect of this assignment, you may end up with more than 1 Climate report or 0 Climate report in a single batch. In the case of the former, you should only select 1 report as the "true" report and drop the other(s). You are free to make the choice on what the "true" report is (i.e., the latest climate report in a batch). For the latter case (no climate report), no processing should occur for that batch.

[2] You can find if two locations are close to each other or not by implementing the geo-hashing algorithm or find a library that does the job for you. The precision parameter in the algorithm determines the number of characters in the Geohash.

[3] You should merge all Hotspot reports which are similar i.e., if there are 3 AQUA reports and 2 TERRA reports with similar location and created time, you should merge all 5 reports into 1.

[4] *First* refers to retaining data from only the first report in a collection in the merged report. Conversely, *last* refers to retaining data from the last report in a collection.

- If a fire was detected with an air temperature greater than 20 (˚C) and a GHI greater than 180 (W/m^2), then report the cause of the fire event as 'natural'. Otherwise, report the cause of the fire event as 'other'.

Save the file as **Assignment_PartB_Streaming_Application.ipynb.**

Task 2 : Data Visualisation (15%)

1. **Streaming data visualisation**
    a. For the incoming climate data plot the line graph of air temperature against arrival time. You need to label some interesting points such as maximum and minimum values.

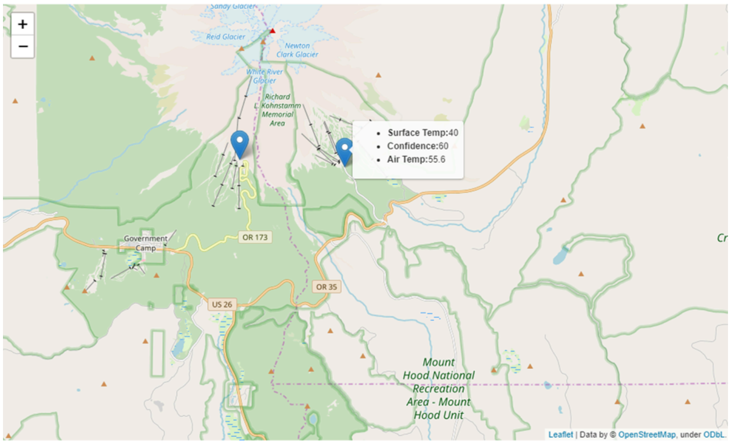
   week 11?

2. **Static data visualisation**

   Write python programs using pymongo to get the data from the MongoDB collection(s) created in Part B, Task 1 and perform the following visualisations.

    a. Plot a bar chart to visualise the total number of fire records based on each hour.

    b. In a map visualise *fire locations* as markers. Use a 'blue' marker if the cause of the fire was *'natural'*. Otherwise, use a 'red' marker. Display detailed information such as *air temperature, surface temperature, relative humidity, and confidence* with the marker tooltip. See the example below. You can use Folium for map visualisation.



- Surface Temp:40
- Confidence:60
- Air Temp:55.6

+
−
Sandy Glacier
Reid Glacier
Newton Clark Glacier
White River Glacier
Richard L. Kohnstamm Memorial Area
Government Camp
OR 173
OR 35
US 26
Mount Hood National Recreation Area - Mount Hood Unit
Cr
Leaflet | Data by © OpenStreetMap, under ODbL.

Save the file as **Assignment_PartB_Data_Visualisation.ipynb.**

Part C (20%)

Task 1: Interview and Demo (20%)

After the assignment due date, you will be asked to attend an interview/demo session to showcase your application. Your interviewer will ask you a few questions in relation to your application and assess your understanding.

Based on your demo and answers to your interview questions, you will receive marks in one of those buckets 20, 15, 10, 5, 0 (Please refer to the marking guide).

Interviews for Assignment-2 will be conducted during the Week 12 lab sessions. If you are granted a short or long extension, the interview will be conducted during SWOT-VAC week.

Assignment Marking

The marking of this assignment is based on the quality of work you have submitted rather than just quantity. Marking starts from zero and goes up based on the tasks you have successfully completed and their quality, for example, how well the code submitted follows *programming standards, code documentation, presentation of the assignment, readability of the code, organisation of code, and so on*. Please find the PEP 8 -- Style Guide for Python Code here for your reference. Please refer to the marking guide for further details.

Submission

This is an individual assignment. You should submit your final version of the assignment solution online via Moodle. You must submit the following:

- A zip file of your Assignment folder, named based on your authcate name (e.g. psan002). This should contain:
  - **Assignment_PartA.ipynb**
  - **Assignment_PartB_Producer1.ipynb**
  - **Assignment_PartB_Producer2.ipynb**
  - **Assignment_PartB_Producer3.ipynb**
  - **Assignment_PartB_Streaming_Application.ipynb**
  - **Assignment_PartB_Data_Visualisation.ipynb**

  This should be a ZIP file and *NOT any other kind of compressed folder (e.g. .rar, .7zip, .tar).*
- The assignment submission should be uploaded and finalised by Tuesday, 21st May 2024, 09:30 AM (MYT).
- Your assignment will be assessed based on the content of the submitted zipped file in Moodle. We will use the same docker image as provided in this unit when marking your assignments. We will also use the same dataset provided to you in this assignment. **If you used additional libraries, please include the pip commands in your Jupyter notebook.**

Resources

FireData: https://sentinel.ga.gov.au/#/
WeatherData: http://www.bom.gov.au/vic/?ref=hdr
Edward, Shakuntala Gupta, and Navin Sabharwal. *Practical MongoDB: Architecting, Developing, and Administering MongoDB*. Apress, 2015.
https://docs.mongodb.com/tutorials/

Other Information

Where to get help

You can ask questions about the assignment on the Ed forum. This is the preferred venue for assignment clarification-type questions. You should check this forum regularly, as the responses of the teaching staff are *official* and can constitute amendments or additions to the assignment specification. Also, you can attend a consultation session with the tutors if the problems and confusion are still not solved.

Plagiarism and collusion

Plagiarism and collusion are serious academic offenses at Monash University. Students must not share their work with any student. Students should consult the policy linked below for more information.

https://www.monash.edu/students/academic/policies/academic-integrity

See also the video linked on the Moodle page under the Assignment block.

The submitted notebook files will be checked for collusion or plagiarism. Students suspected of colluding or plagiarising the assignment will be reported to the Student Conduct and Complaints Department for academic misconduct. Consequently, your grade for this unit will be withheld until the investigation is complete.

Generative AI usage

Generative AI tools cannot be used in this assessment task. In this assessment, you must not use generative artificial intelligence (AI) to generate any materials or content in relation to the assessment task. Students are required to produce their own solutions in this assignment. In addition, using generative AI may provide inaccurate outcomes and, therefore, is prohibited.

Late submissions

Extensions and other individual alterations to the assessment regime will only be considered using the University's Special Consideration Policy.

There is a **10% penalty per day, including weekends,** for late submission.