Assignment

Data Engineering for Distributed Micro-Service Pipeline

Due Date: Wednesday, February 19, 2020 End of Day

Group Demonstration: Friday, February 21, 2020 - 8:30 to 11:00 IT 101

The assignment will test the ability of students to create a distributed analytics pipeline by applying their data engineering and analytics skills and knowledge. The assignment will provide an opportunity for students to learn how to design and implement distributed pipelines for AI.

Task – Real-time Feedback for Taxi Drivers in NYC

The students should design and implement a pipeline that streams, preprocesses, cleans, enriches, and transforms data items in order to enable real-time feedback to taxi drivers regarding peak times and busy locations. The pipeline can be developed in Python or Java. The pipeline should be designed in a scalable manner following service-oriented and micro-service design principles.

Datasets

Yellow Taxi Trip Records (January 2018)

The dataset provided contains yellow taxi trip records capturing pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts. The data used in the dataset were collected and provided to the NYC Taxi and Limousine Commission (TLC) by technology providers authorized under the Taxicab and Livery Passenger Enhancement Programs (TPEP/LPEP).

TLC URL: **https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page**

Dataset URL: https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2018-01.csv

Taxi Zone Lookup Table: https://s3.amazonaws.com/nyc-tlc/misc/taxi+_zone_lookup.csv

Yellow Trip Data Dictionary:
https://www1.nyc.gov/assets/tlc/downloads/pdf/data_dictionary_trip_records_yellow.pdf

Motor Vehicle Collisions - Crashes

The Motor Vehicle Collisions crash table contains information from all police-reported motor vehicle collisions in NYC. Each row represents a crash event.

URL: https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95

Pipeline Input: The students should stream the NYC Taxi dataset into their system. This requires them to load the dataset and simulate a stream of records as individual taxi trip updates.

The accident data set should be used as a source to enrich the data.

Pipeline Time: The pipeline should process data in a continuous manner and the publisher should not adhere to the time-delay mentioned in the event timestamps when streaming the data. However, the timestamps should be used to determine the windows required for the analysis. (i.e. use the event timestamp to determine when a window has elapsed)

Pipeline Output: The pipeline should collect individual trip records to perform the following analysis:

- Implement daily and hourly tumbling windows based on Dates/Times in input records
- Calculate the frequency of taxi trips within specified time periods
- Compare results and identify peak times and busiest locations (boroughs and zones)
- Report on accidents at the identified peak times and locations within the specified time periods
- Results should be emitted using an appropriate representation continuously both hourly and daily.

Pipeline Design: The system should be designed in a manner consistent with service-oriented design principles to enable deployment with a XaaS model. The system should be comprised of multiple operators, each encapsulated as a stand-alone microservice. The system should process records as they arrive in a sequential manner (one trip at a time). The design should decompose the system into logical entities, each responsible for a specific role (e.g. Analytics, etc).

All services should communicate and orchestrate the data flow through a messaging substrate, in the form of a stand-lone service that implements a suitable inter-service communication mechanism (i.e message-driven, publish/subscribe). Note that students are not responsible for developing this communication mechanism and can use an existing tool. We recommend the use of Apache ActiveMQ which is an open-source, multi-protocol, java-based messaging service that also supports connectivity through a number of other programming languages including python.

Code (60 Marks)

- Preprocessing (i.e. mapping, cleaning, denormalization, etc) (15)
- Data Enrichment (10)
- Analytics and state management (10)
- Output result and representation (5)
- Pipeline scalability and optimization techniques (20)

Report Format (40 Marks)

Please submit a short report (approximately 4-5 pages in addition to your code and references) specifying:

- **Pipeline Design:** A description of your approach and design decisions. This should include details on the flow of the pipeline, inter-service communication mechanism, the logic for separation of concerns. Justify your design decisions. Where appropriate include including appropriate diagrams and references to research papers to support your arguments.
- **Pipeline Output:** Detail the output of your pipeline.
- **Design Strengths and Weaknesses:** Detail the Strengthens and Weaknesses of your approach Justify your design decisions. Where appropriate include references to research papers to support your arguments. (minimum 1 Page)
- **Scalability:** Discuss the scalability of your pipeline and the techniques you have leveraged to optimize it. Include any references to research papers to justify your choices (minimum 1 Page)

The report should include your name, class and student numbers. Your code, attached as an appendix. (Code and references are covered in the page count)

Groups

This is a group assignment. Each group should have 2 students. Groups are self-selected. Please email your selection to Tarek at tarek.zaarour@insight-centre.org by Friday 24th January. If you cannot form a group, please inform Tarek.

Submission Instructions

- Please put your code into a single .zip archive with name "YourNames_ResearchTopic1_code.zip", submit via Blackboard
- Include any file or component which is needed to re-run the assignment such as scripts and instructions. Please do not include any database internal files.
- Include all source code files (that is, files with name ending .java etc) required to compile and run your code.
- Include a screenshot of the output of your application for any relevant output.
- Use comments to explain your source code. Insufficient comments can lead to mark deductions.
- Please put your report into a single .pdf file with name "YourNames_CaseStudyAssignment1_report.pdf", submit via Blackboard
- Please note that all submissions (both code and report) will be checked for plagiarism. Plagiarism (from another group or other sources) are not allowed and will be treated seriously.

Common Questions on Assignment

Q: The records for the taxi trips are Out Of Order (OOO).
A: There are two options here. 1) Sort the data outside of your solution or 2) enable support for out of order messages in your solution. Option 1 is the simplest and we would

recommend you start with this approach. Option 2 is more advanced and would gain extra marks. Detail the consequences of the choices in the report.

Q: Should I use a framework (i.e. ESPER) to support the windows?
A: The logic for the windowing is relatively straight forward. A framework may take more time to setup than a simple implementation. Justify your design choices in the report.

Q: Should I use a Database for state management?
A: Depends on the goal, if its to persist data for later analysis, using a database might be beneficial but it can also be simply done using files. If the goal is to support the real-time processing, it might add overhead, so proceed with caution. Justify your decision in the report.