W205 – PROJECT 1 – FOOD NINJA RECOMMENDATION ENGINE - SRIRAM RAO - ARCHITECTURE

Project Objective

Design and Develop an end-to-end data processing pipeline for a real-world business scenario with a production dataset based on concepts learnt from UCB-W205 Storing and Retrieving Data course. More specifically:

- Use HDFS to host a large volume (750 MB) dataset for ingestion
- Use Apache Spark with PySpark and Spark-SQL to process the data in to business insights
- Use Postgres DB to store the processed information for lower latency
- Build a REST API based serving layer that can invoke either the Spark processing job or obtain data from a database
- Implement the data processing pipeline in a cloud-based server infrastructure

Business Use Case

A. Business and leisure travelers are always looking for:

    1. "Which is the best restaurant (based on user reviews) to get some <Pancakes> around here"
    2. "So my team is going to this restaurant, wonder what dish is good out there (based on user reviews)"

B. If a restaurant is looking to expand or grow:

    1. "Our <food category> restaurant is taking off, which other cities should we consider to open another franchise"

Dataset

The primary dataset is Yelp's challenge dataset. You can download the data from https://www.yelp.com/dataset_challenge

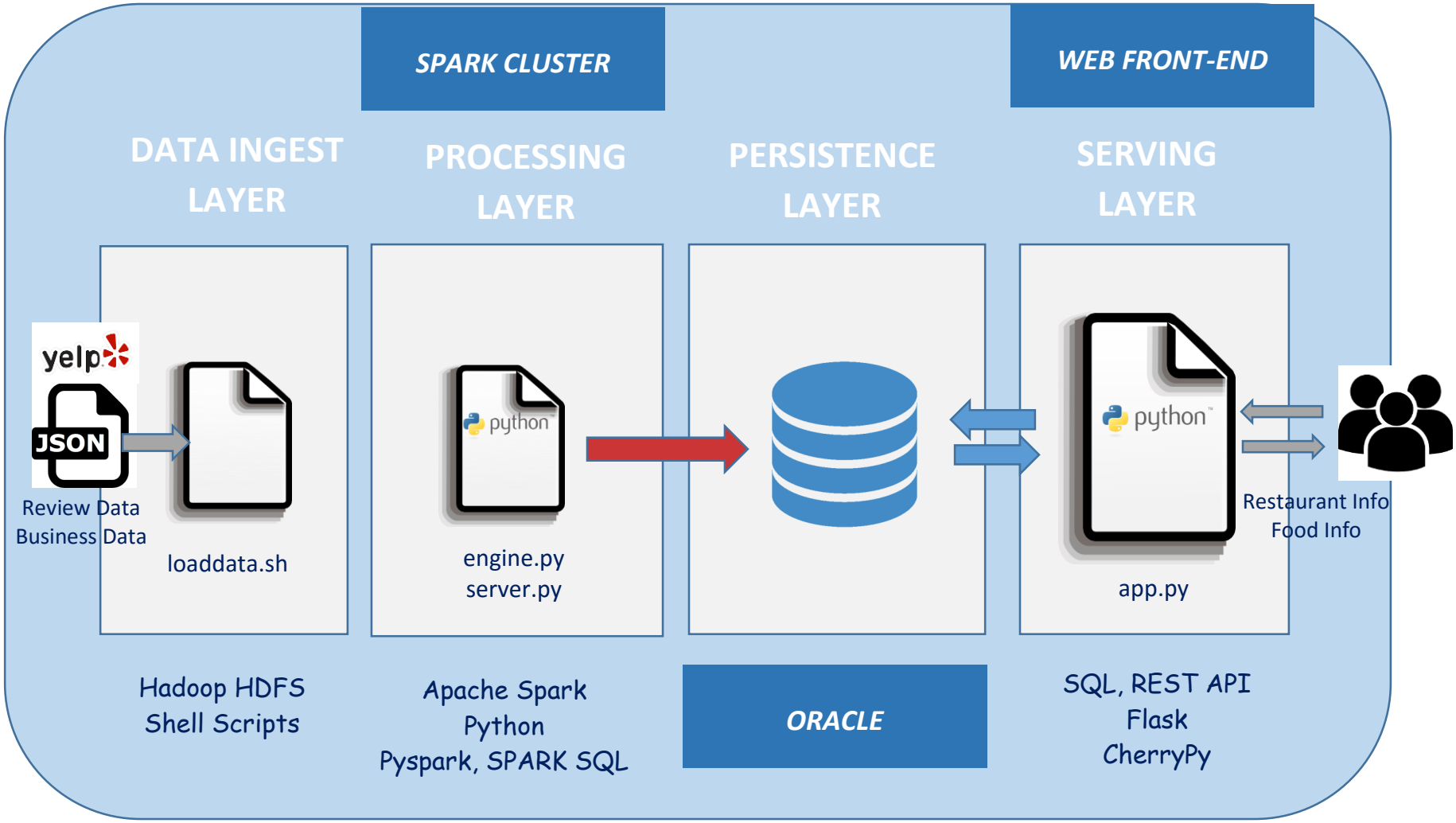
1. Food Review Data - Yelp dataset (1.8 GB):
    - 2.2M reviews and 591K tips by 552K users for 77K businesses
    - Social network of 552K users for a total of 3.5M social edges.
    - Aggregated check-ins over time for each of the 77K businesses
2. Business Data – Yelp dataset (65 MB)
    - Data for 77K businesses
    - 566K business attributes, e.g., hours, parking availability, ambience.

Production Architecture

Below is schematic of the **proposed** Production Architecture of the end-to-end data processing pipeline application with a clearly defined data ingest layer, processing layer and data persistence layer.

- Data Ingest Layer: In the production environment, it is expected that the business data and the review data will be refreshed on an hourly or at the least daily schedule (VELOCITY). The data VOLUME is also expected to be large. The test dataset used in this Proof of Concept has data for about 300 medium and small cities. The zipped dataset size for the 300 cities is 750 MB. This is expected to be in the several hundred GB range in Production. The data ingest layer therefore has to be robust. We propose a HDFS storage layer for the data that will be ingested by the data pipeline.
- Data Processing layer: The data pipeline is being configured to handle hundreds of requests that can processed by multiple workers in Production. Apache Spark with its versatile feature set that allows request jobs to be run with python, java and SQL is the most appropriate solution. The processing layer must also be run within a clustered environment.
- Data Persistence Layer: For fast (sub 2 seconds) responses, all known combinations of data inputs will processed in advance and materialized in a database. Based on our initial analysis, the application is expected handle about 50 concurrent requests a second. To support this concurrency with fast response times, the Spark responses will be cached in an Oracle database.
- Serving Layer: The requests for the food ninja food recommendation engine is expected from several third-party applications. REST API with its powerful yet simple interfaces provides the fastest option to serve the end users. The REST API will be architected with CherryPy and flask.

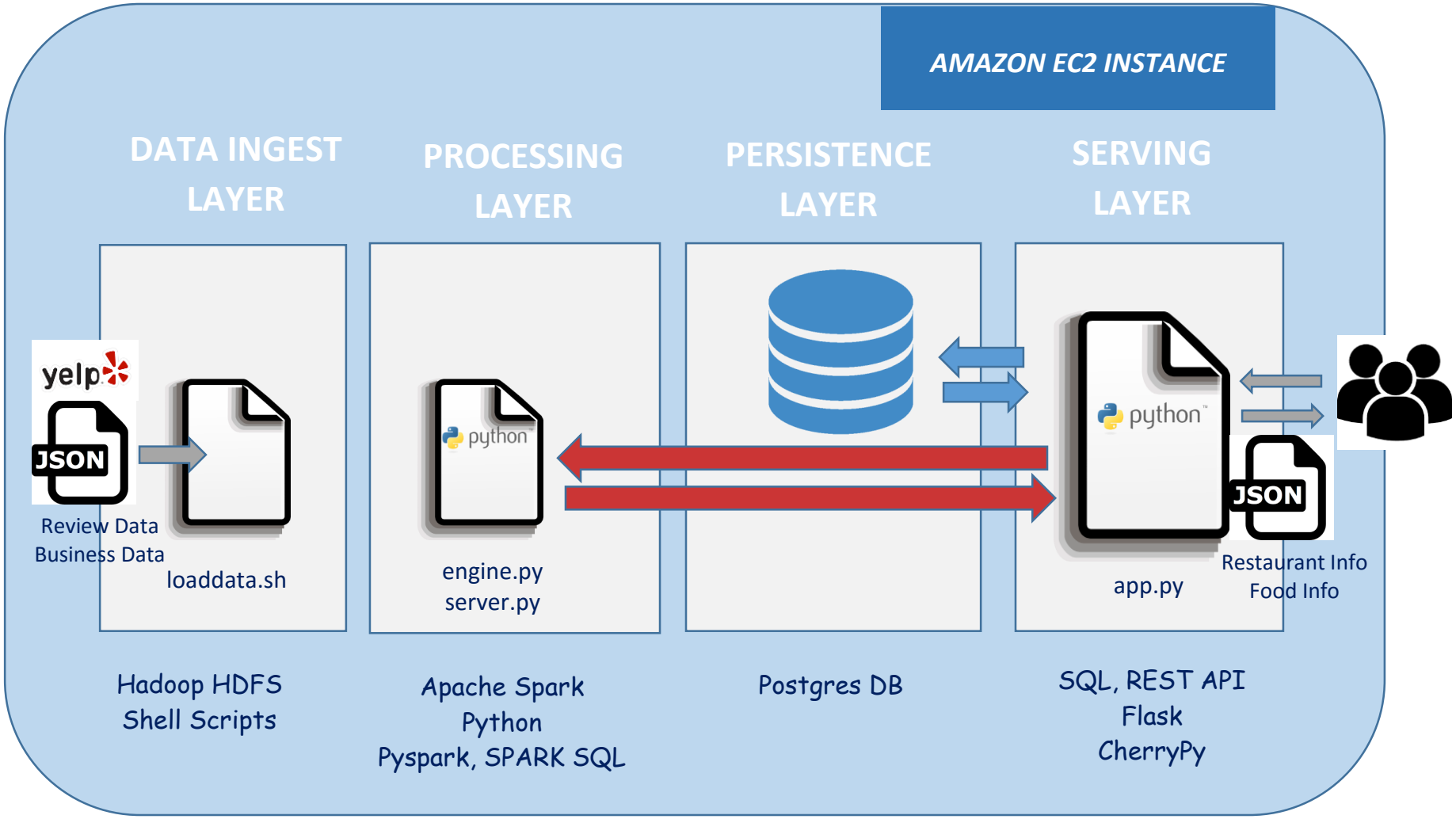
W205 – PROJECT 1 – FOOD NINJA RECOMMENDATION ENGINE - SRIRAM RAO - ARCHITECTURE



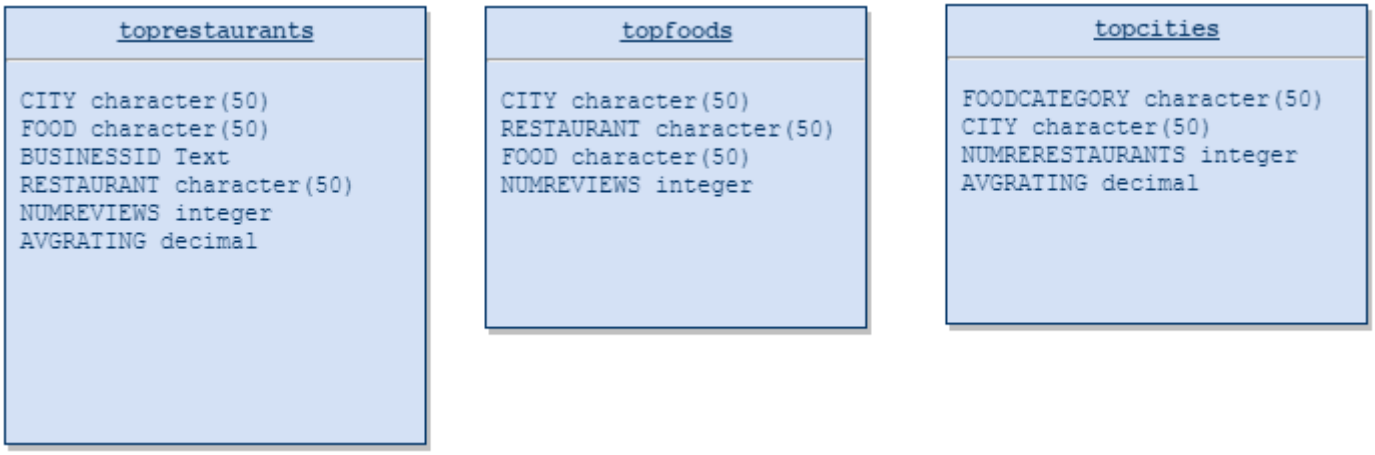
Proof of Concept Architecture

This proof of concept is being developed to quickly demonstrate the feasibility to provide a food recommendation engine with Yelp data. Therefore we have made a few changes to the Production Architecture described above. For the most part the application will follow the layered architecture described above. However the following changes have been made:

- The data pipeline is executed in a single EC2 instance
- The data is persisted in a Postgres database
- When a request is received for the first time from a REST API call, it triggers a pyspark/Spark-sql batch job. The results of that job will be stored in the database for subsequent requests. Therefore initial requests are expected to be slow. Similar requests made later will be served from the database and are therefore expected to be quicker.
- The proof of concept solution has been developed for the following three REST API calls:
    - Restaurant Search with City and Food as input parameters: /restsearch/<string:city>/<string:food>
    - Food Search with City and Restaurant as input parameters: /foodsearch/<string:city>/<string:restaurant>
    - City Search with Food category as input parameter: /citysearch/<string:category>


AMAZON EC2 INSTANCE
DATA INGEST LAYER
PROCESSING LAYER
PERSISTENCE LAYER
SERVING LAYER
yelp
JSON
Review Data
Business Data
loaddata.sh
python
engine.py
server.py
python
app.py
JSON
Restaurant Info
Food Info
Hadoop HDFS
Shell Scripts
Apache Spark
Python
Pyspark, SPARK SQL
Postgres DB
SQL, REST API
Flask
CherryPy

Data Model


toprestaurants
CITY character(50)
FOOD character(50)
BUSINESSID Text
RESTAURANT character(50)
NUMREVIEWS integer
AVGRATING decimal
topfoods
CITY character(50)
RESTAURANT character(50)
FOOD character(50)
NUMREVIEWS integer
topcities
FOODCATEGORY character(50)
CITY character(50)
NUMRERESTAURANTS integer
AVGRATING decimal