Project 1: Big Data Programming Paradigms

Cloud computing and Big Data represent technically different terms, but they are often seen together because of the strong interaction between them. While Big Data simply refers to the capacity to deal with a large amount of data using parallel paradigms, cloud computing usually refers to the processing of anything, including Big Data programs. The cloud, however, provides the compute and storage resources needed to process large amounts of data using parallel paradigms. The cloud provides access to computational resources previously unavailable to many organizations.

In this project, students will get experience with Spark, one of the most popular Big Data frameworks that have been adopted for use in cloud systems to date.

Introduction

Streaming boosts music consumption by offering on-demand access to a large music collection from any connected device, anytime, and anywhere through playlists. Playlists are often used as mechanisms of musical discovery, collection and identity formation, being one of the main changes fostered by streaming. In this sense, Spotify is one of the most popular music streaming services in the world with more than 80 million songs and more than 2 billion playlists created manually by users or dynamically by the platform based on music recommendation systems. Platforms such as Spotify, which provide an API for queries, have been used in several studies [1, 2]. The study of music consumption is important in several areas of science because it generally reflects the language, feelings and behavior of users and the cultural environment around them. In computer science, for instance, it helps to improve the state of the art in recommendation systems.

Dataset Description

The dataset, tools, and frameworks needed to complete the assignment are already installed in the cluster.

The dataset is in HDFS at `hdfs://localhost:9000/datasets/spotify/`. All students have read access to the dataset. The tasks in this project do not require writing any results in files; however, students can use their storage space in HDFS at `hdfs://localhost:9000/user/<netid>` to store any partial result that they want, if necessary.

> Careful: Do *not* duplicate the dataset while completing the assignment. This would quickly fill all available disk space and make the cluster unusable.

The database used represents an extraction of the lists of songs heard on Spotify, obtained by The Million Song Dataset. The original base contains 1.6 million playlists created by users on

the Spotify platform, and was collected by researchers interested in exploring how to improve the music listening experience. These playlists were created during the period from January 2010 to October 2017. This dataset is divided into two JSON files, the first (`playlists.json`) containing playlist metadata such as the name, the last modification date and the number of followers while the second (`tracks.json`) containing information about more than 10 million songs present in the playlists. The datasets total approximately 3.0 GB of data. More specifically, each JSON object representing a playlist (in `playlists.json`) contains the following fields:

- `pid` : the ID of the playlist;
- `name` : the name of the playlist;
- `modified_at` : timestamp (UNIX epoch) of when this playlist was last updated;
- `duration_ms` : the total duration of all tracks in the playlist (in milliseconds);
- `num_albums` : the number of unique albums for the tracks in the playlist;
- `num_followers` : the number of followers this playlist had at the time the database was created;
- `num_artists` : the total number of unique artists for the tracks in the playlist;
- `num_edits` : the number of distinct edit sessions;
- `num_tracks` : the number of tracks in the playlist.

Each JSON object representing a track (in `tracks.json`) contains the following fields:

- `pid_playlist` : the ID of the playlist containing the track;
- `track_name` : the name of the track (title);
- `track_uri` : the Spotify URI of the track, this field can be used as a key to identify individual songs in the dataset (in other words, each track has a single unique URI)
- `album_name` : the name of the track's album;
- `album_uri` : the Spotify URI of the album;
- `artist_name` : the name of the main artist of the track;
- `artist_uri` : the Spotify URI of the track's lead artist;
- `duration_ms` : the duration of the track in milliseconds;
- `pos` - the position of the track in the playlist (zero-based);

Tasks

In this project, you are tasked with answering some questions about music consumption on the Spotify platform. More specifically, you will perform the following analyses.

1. Statistics about songs duration

On Spotify, there are albums with short tracks (like transitions between songs on an album or advertising) but there also exist songs that exceed 1 hour (for example, full concerts or mixes). These types of tracks can be considered an outlier.

> An outlier is a data point that goes far outside the average value of a group of statistics. Outliers may be exceptions that stand outside individual samples of populations as well.

A simple way to remove outliers is using an Interquartile Range Rule (IQRR), a technique that removes points outside an interval defined by the 1st and 3rd quartiles. In this task, you will perform the following steps:

1. Generate a table containing the minimum, average and maximum duration, in milliseconds, of the songs in the dataset.
2. Compute the first and third quartiles (denoted Q_1 and Q_3), as well as the interquartile range (IRQ) ($Q_3 - Q_1$).

   > In statistics, the first quartile (Q1) is defined as the smallest value larger than 25% of the sample, while the third quartile (Q3) is the smallest value larger than 75% of the sample. Both quantiles are used to compute the IQR metric, defined by $Q_3 - Q_1$.

3. Compute the set of songs with durations that are not outliers, as defined by the IQRR methodology. In other words, identify all songs with duration x such that $Q_1 - 1.5 \times \text{IQR} < x < Q_3 + 1.5 \times \text{IQR}$.
4. Using the IQRR methodology, how many songs would be considered outliers and removed from analysis? Generate a new table containing the minimum, average and maximum duration of the remaining songs.

Because the data is well-structured, Spark's DataFrame is a good and well-documented interface, with a set of operators to handle Big Data in Spark. The DataFrame abstraction offers operators like `read.json`, `filter`, `groupby`, and `join` that can be used in this exercise. Spark parallelizes the execution of these operators, making computation significantly faster than on a single-thread program (as one would get when using Pandas). This tutorial provides an introduction to PySpark DataFrames and contains several examples.

2. Finding the most popular artists over time

Finding popular artists can be interesting to analyze user tendencies and to inform organizations preparing advertising campaigns. In this task, find the five most popular artists ranked by the number of playlists they appear in. Create a chart that shows the number of playlists containing each of these five artists over the years. Consider that an artist is present in a playlist after each playlist's last modification date.

> The `matplotlib` library supports building rich graphs directly from Python. This page

> provides instructions on how to plot line graphs, as needed in this task.

3. Playlists's behavior

Playlist to collect different songs by user preference, musical genre, or a variety of other relationships. In this sense, your task is analyzing how playlists are being created. What is more common, playlists where there are many songs by the same artist or playlists with more diverse songs? To answer this question, compute the *prevalence* of the most frequent artist in each playlist, defined as the fraction of songs by the most frequent artist. Then create a Cumulative Distribution Function (CDF) plot containing the distribution of artist prevalence across all playlists.

> The Cumulative Distribution Function (CDF), of a real-valued random variable $\mathcal{V}$, evaluated at x, is the probability function that $\mathcal{V}$ will take a value less than or equal to x. CDFs are used to describe the probability distribution of random variables.

Distributed Computing

For the purposes of this course, you should use Spark's distributed computing interfaces. To not fall back to writing plain Python or to using frameworks that are not distributed. You should not use Pandas, and should not do any heavyweight processing using libraries like NumPy and SciPy (using NumPy for handling processed (small) data when plotting the graphs is OK). When in doubt, ask the instructors.

Grading

This assignment is worth 0.125 points, and all tasks will be graded with equal weight: each task is worth 0.047 points. Graphs should be understandable given only a short caption: Graphs should include a title and have a label on each axis. Graphs should also include a legend when appropriate (for example, when there are multiple lines in a graph).

Although not a strict requirement, students should strive to make efficient use of computing resources on the cluster. Remember that the cluster is shared among all students, be careful not to extrapolate its resources.

What to Submit

You should submit:

1. All code you developed in this project. Organize the code of each task in a separate directory and submit a zip file containing code for all tasks.
2. A PDF file containing:
    i. Two tables and a paragraph discussing the results for Task 1.

ii. One graph and a paragraph discussing the results for Task 2.
iii. One graph and a paragraph discussing the results for Task 3.