Spotify Recommendation Platform

(Anand Rastogi – 2022OG04054)

Problem: Spotify Music Recommendation Platform

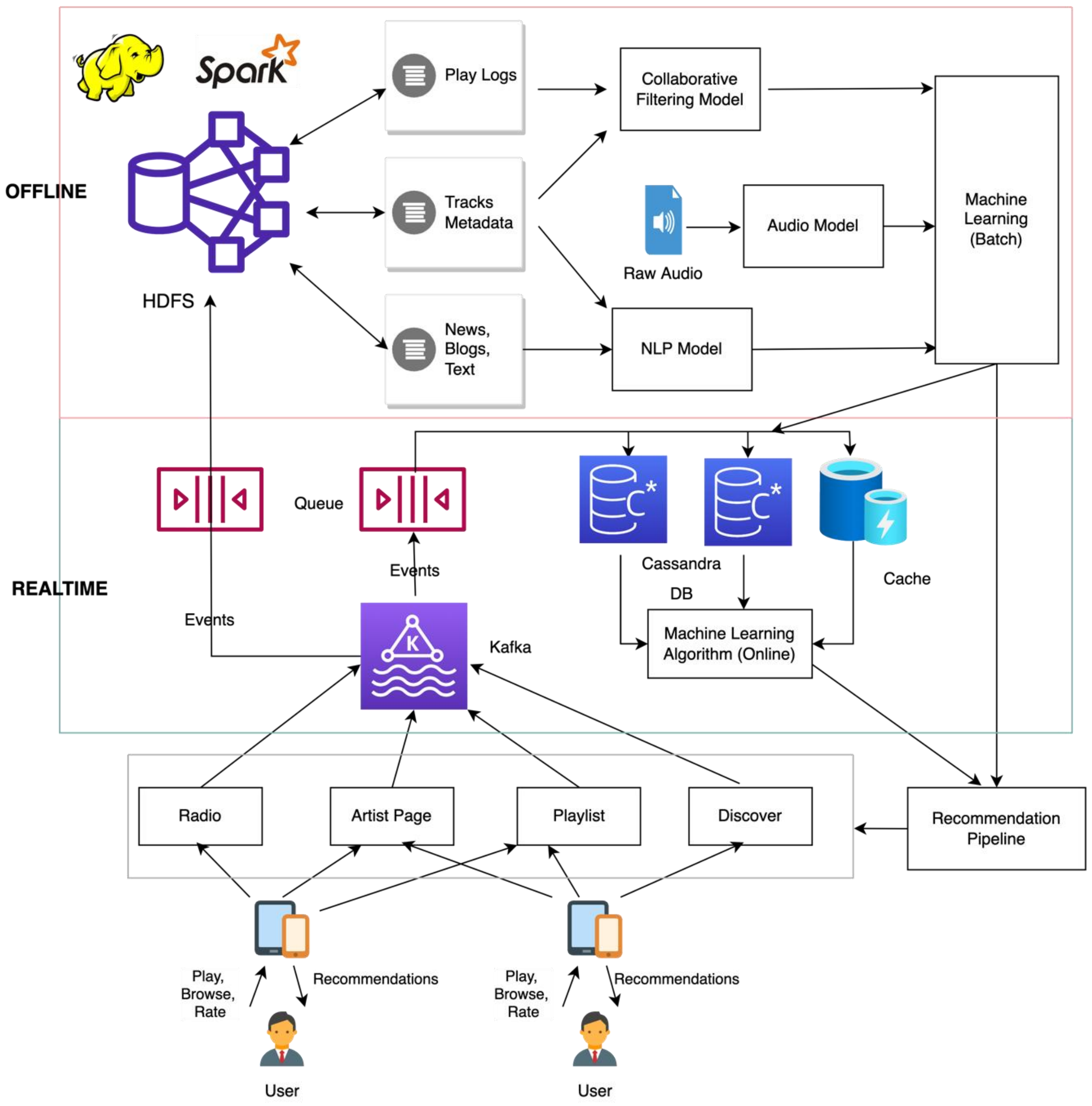
With multiple audio features such as danceability, energy, and valence, this dataset can be used to build a music recommendation system. By analyzing the preferences of users for certain genres and characteristics of tracks, the system can suggest similar tracks or even recommend new genres that users might enjoy.

End-User experience: As a user searches or plays a particular song in Spotify, your Music Recommendation Platform will be able to return other music recommendations for the user. And if the user selects or ignores the recommendation, that data is fed back to improve the recommendation algorithm.

1) **Design:** Design the backend services, APIs, data models and the databases you will use for this. The design should support both real-time stream processing and batch analytics use cases. Explain the rationale for your design especially tradeoffs w.r.t CAP theorem, describe the tech choices for the databases and other components. You must use a NoSQL database as part of your design (MongoDB, Neo4j, DynamoDB, HBase, anything appropriate for your solution).

Solution notes: Submission for this part MUST include an Architecture diagram with bullet points or a brief description of the design choices, rationale, tech choices.

GitHub Link - https://github.com/AndyBITS/BDS_Assignment_SpotifyRecommendations.git



1) Spotify Music Recommendation Platforms will comprise processing of real time events in Realtime as well as Offline mode.
2) User plays, browses and rates the songs from Spotify access points such as Apps, and WebUI etc. Every song browsed, played or skipped would generate an event which will be sent through Kafka streaming layer.
3) Realtime processing of events will enhance the user's experience by providing the recommendations based on recent activities such as Continue Playing, Recently Listened Songs, Artists Playlist and Genre.

4) Offline processing of events will enhance the user's experience by providing the recommendations from ML Models implemented on aggregated data over a long period.
    a. Collaborative Filtering Model: Recommendations based on similarity and behavior identified among users.
    b. NLP Models: Recommendations on news articles and blogs published for artists, music tracks and albums.
    c. Content Filtering Model: Recommendations based on insights gained by analysis of songs metadata.
    d. Audio Filtering: Recommendations based on raw audio analysis to get the insights on danceability, energy and valence.

Infrastructure Design/Proposal

1) Kafka is used for streaming the events.
2) Cassandra (NoSQL) DB is proposed for real-time processing mode.
    a. Aligns with Availability and Partition Tolerance principles in CAP Theorem.
    b. Scalability: It ensures highly availability by replicating data within and across the nodes.
    c. Read Optimized: Cassandra is based on Query First Approach, and we can design the tables based on the queries we are going to design. The data will be loaded in partitions spread across the nodes, enabling the fast retrieval. E.g.
        i. The data with different genres, artists will be loaded in different nodes and help with fast retrievals with simple queries (not requiring join operations).
        ii. The write/update on specific genre will not lock the partitions with another genre.
    d. Eventual Consistency: The data consistency requirement is not immediate in music recommendation platform ad nwe should be good with near time latency for data to be updated across replicas in all nodes.
    e. Cache has also been enabled to handle the read/write latency from ML algorithms.
    f. Cassandra is scalable and we can add more nodes whenever we need more data to be stored.
3) Hadoop Cluster is proposed for offline processing mode to store the logs, events and existing metadata for tracks. Spark can be used for processing.

4) Machine Learning Algorithms can be implemented using implemented through Spark.

2) **OLTP Queries:** Execute at least 3 key read/write queries against the NoSQL database as part of the solution with the optimizations and the consistency levels set in your queries.

Solution notes: Submission for this part MUST include a pdf of the queries and a video of their successful execution (you can use loom for the video). No more than 2 pages for the queries and no more than 1 minute for the loom video.

Since we are designing the music recommendation platform and Cassandra follows a query first approach model, our select queries will be mainly used for retrieving and recommending the music tracks on features such as track_genre, artists, and popularity etc. Hence, we should have the tables created with these attributes as the partition keys for optimized results.

Since Cassandra was installed with one node on personal laptop,I could not capture the instances with consistency levels other than One. I did try to change level to Any/Quorum once but I started getting error notifications.

Recording link -
https://www.loom.com/share/e3a116ac1fc04341ad90698f3180cb7d?sid=c6e9457d-1359-4a8d-907b-ee94aca6e6d3

Below are the sample READ/WRITE queries we will execute as part of this assignment:

a. Retrieve the 10 most popular songs for a specific artist.

```
 cqlsh:music_keyspace> select track_name, popularity, artists from
music_by_artists_popularity where artists = 'Acid Ghost' order by
popularity desc limit 10;
 track_name             | popularity | artists
------------------------+------------+------------
              Sandy Kim |         57 | Acid Ghost
        I Don't Need You |         53 | Acid Ghost
               New York |         52 | Acid Ghost
           Overthinking |         50 | Acid Ghost
              All Alone |         49 | Acid Ghost
    Not the Same Person |         49 | Acid Ghost
          Nature (Poem) |         48 | Acid Ghost
           Hide My Face |         45 | Acid Ghost
```

```
(8 rows)
cqlsh:music_keyspace>
```

b. **Retrieve any10 songs, album and artist for a specific track_genre.**

```
cqlsh:music_keyspace> select track_name, album_name, artists from
music_by_track_genre where track_genre ='rock' limit 10;

 track_name                        |
album_name                          | artists
--------------------------------+-----------------------------------
+------------------------------
                 Mr. Brightside |                          Hot Fuss
|                    The Killers
                Merry Christmas |             Christmas Dinner 2022
|                    Bryan Adams
 Little Saint Nick - 1991 Remix | Best Christmas Playlist Ever 2022
|                 The Beach Boys
           I Put A Spell On You |                Pumpkin Patch Hits
| Creedence Clearwater Revival
                 Summer Of '69 |                   Best Of Karneval
|                    Bryan Adams
          All My Favorite Songs |   Here to Forever - All New Indie
|                     Weezer;AJR
 There’d Better Be A Mirrorball |  I Ain't Quite Where I Think I Am
|                 Arctic Monkeys
          Christmas Without You |             CHRISTMAS TOP HITS 2022
|                    OneRepublic
                      Centuries |   American Beauty/American Psycho
|                   Fall Out Boy
                 Blood // Water |               All Out Alternative
|                       grandson
```

c. **Insert a new track for a new artist and retrieve the row to confirm.**

```
cqlsh:music_keyspace> consistency one
Consistency level set to ONE.

cqlsh:music_keyspace> insert into music_by_artists_popularity
(artists,
popularity,track_id,album_name,track_name,duration_ms,explicit,dance
ability,energy,key,loudness,mode,speechiness,acousticness,instrument
alness,liveness,valence,tempo,time_signature,track_genre) VALUES
('Ghulam Ali',100,'1ne2wOtXF2jM6Cm8WBkaER','Hungama','Hungama Hein
Kyun Barpa',55555, FALSE,0.421,0.637,3,-
1000,1,0.01,0.1,0,1,1,100,4,'Retro Ghazal');

cqlsh:music_keyspace> select * from music_by_artists_popularity
where artists = 'Ghulam Ali';

 artists    | popularity | track_id               | acousticness |
album_name | danceability | duration_ms | energy | explicit |
instrumentalness | key | liveness | loudness | mode | speechiness |
tempo | time_signature | track_genre | track_name              |
valence
------------+------------+-------------------------+--------------+--
----------+--------------+-------------+--------+----------+--------
----------+-----+----------+----------+------+------------+-------
+----------------+-------------+-------------------------+---------
 Ghulam Ali |        100 | 1ne2wOtXF2jM6Cm8WBkaER |          0.1
|    Hungama |         0.421 |        55555 |  0.637 |     False
|                 0 |    3 |         1 |    -1000 |     1 |         0.01
|   100 |                 4 |      Ghazal | Hungama Hein Kyun Barpa
|       1

(1 rows)
```

d. Update the track_genre for an existing artist.

```
cqlsh:music_keyspace> consistency
Current consistency level is ONE.


cqlsh:music_keyspace> update music_by_artists_popularity set
track_genre = 'Ghazals' where artists='1349' and popularity = 20 and
track_id = '0Js35d0JcXFyXnVQgzrYx1';

cqlsh:music_keyspace> select * from music_by_artists_popularity
where artists = '1349' and popularity = 20 and track_id =
'1ne2wOtXF2jM6Cm8WBkaER';
;

 artists | popularity | track_id               | acousticness |
album_name           | danceability | duration_ms | energy |
explicit | instrumentalness | key | liveness | loudness | mode |
speechiness | tempo      | time_signature | track_genre |
track_name                 | valence
---------+------------+------------------------+-------------+-----
-----------------+--------------+-------------+--------+----------+-
-----------------+-----+----------+----------+------+-------------+-
----------+----------------+-------------+-------------------------
+---------
    1349 |         16 | 54HlHAstsrNh79FNGme31K |      2.2e-05 | The
Infernal Pathway |         0.203 |      489333 |  0.928 |     False
|             0.699 |    9 |     0.307 |  -10.858 |     0 |       0.0866
|   119.634 |                4 | black-metal |        Stand Tall in
Fire |   0.195
    1349 |         17 | 4HpawkkZ4NYYaWMIHctf3x |      3.3e-05
|             Hellfire |         0.318 |      829146 |  0.999
|    False |            0.795 |    4 |     0.201 |   -6.346 |    0
|       0.207 | 134.81599 |                4 | black-metal
|                  Hellfire |  0.0225
    1349 |         17 | 5C6oM7CRpTXJTnG0iQvXuj |      5.8e-05
|           Liberation |         0.163 |      274640 |  0.993
|    False |            0.552 |    4 |     0.381 |   -4.106 |    1
|       0.147 |   147.591 |              4 | black-metal | Riders Of
The Apocalypse |   0.236
    1349 |          20 | 0Js35d0JcXFyXnVQgzrYx1 |       2.3e-06
|             Hellfire |         0.169 |       465586 |  0.981
|    False |             0.788 |    0 |     0.303 |    -5.07 |    1
|       0.133 |     117.6 |              4 |     Ghazals | Celestial
Deconstruction |  0.0393
    1349 |          20 | 60edNIKZXcOi95A3BcNIms |       4.3e-06 | The
Infernal Pathway |        0.0998 |       330066 |  0.911 |     False
|           0.714 |    9 |      0.18 |  -11.959 |    0 |       0.112
|   195.039 |              4 | black-metal |        Abyssos
Antithesis |  0.0708

(5 rows)
```

3) **OLAP Analytics Queries:** Within the broader context of this problem (Spotify) you need to perform analytics queries from the dataset to driver better Sales / Growth. For this, setup a big data batch processing environment (Hadoop or a Spark), share the config. Then perform at least 3 queries in the form of Map-Reduce jobs or Pig / HiveQL / SparkSQL queries and share the responses.

Solution notes: Submission for this part MUST include a pdf of the environment configuration, the queries and a video of their successful execution (you can use loom for the video). No more than 2 pages for the config and queries and no more than 1 minute for the loom video.

Refer the files available at GitHub link - .

https://github.com/AndyBITS/BDS_Assignment_SpotifyRecommendations/blob/main/BDS%20Assignment_Spotify%20recommendations_Config_Anand%20Rastogi_2022OG04054.html

https://github.com/AndyBITS/BDS_Assignment_SpotifyRecommendations/blob/main/BDS%20Assignment_Spotify%20recommendations_OLAP%20Analytics_Anand%20Rastogi_2022OG04054.html

BDS Assignment_Spotify

BDS Assignment_Spotify

4) **Appendix**

DDL/ DML SQL

```
cqlsh> CREATE KEYSPACE music_keyspace WITH replication =
{'class':'SimpleStrategy','replication_factor':3} AND
durable_writes = 'true';

Warnings :
Your replication factor 3 for keyspace music_keyspace is
higher than the number of nodes 1.


cqlsh> USE music_keyspace;
cqlsh:music_keyspace>


cqlsh:music_keyspace> CREATE TABLE music_by_track_id (track_id
text PRIMARY KEY, artists text, album_name text, track_name
text, popularity int, duration_ms int, explicit boolean,
danceability float, energy float,key int,loudness float,mode
int,speechiness float,acousticness float,instrumentalness
float,liveness float,valence float,tempo float,time_signature
int,track_genre text);
cqlsh:music_keyspace> describe table music_by_track_id

CREATE TABLE music_keyspace.music_by_track_id (
    track_id text PRIMARY KEY,
    acousticness float,
    album_name text,
    artists text,
    danceability float,
    duration_ms int,
    energy float,
    explicit boolean,
    instrumentalness float,
    key int,
    liveness float,
    loudness float,
    mode int,
    popularity int,
    speechiness float,
    tempo float,
    time_signature int,
    track_genre text,
    track_name text,
    valence float
) WITH additional_write_policy = '99p'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrate
gy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '16', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND memtable = 'default'
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND extensions = {}
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99p';

cqlsh:music_keyspace> select count(*) from music_by_track_id;

 count
-------
 89621

(1 rows)

Warnings :
Aggregation query used without partition key



cqlsh:music_keyspace> CREATE TABLE music_by_artists_popularity
(artists text, popularity int, track_id text , album_name text,
track_name text, duration_ms int, explicit boolean,
danceability float, energy float,key int,loudness float,mode
int,speechiness float,acousticness float,instrumentalness
float,liveness float,valence float,tempo float,time_signature
int,track_genre text,PRIMARY KEY (artists,
popularity,track_id));
cqlsh:music_keyspace> describe table
music_by_artists_popularity

CREATE TABLE music_keyspace.music_by_artists_popularity (
    artists text,
    popularity int,
    track_id text,
    acousticness float,
    album_name text,
    danceability float,
    duration_ms int,
    energy float,
    explicit boolean,
    instrumentalness float,
    key int,
    liveness float,
    loudness float,
    mode int,
    speechiness float,
    tempo float,
    time_signature int,
    track_genre text,
    track_name text,
    valence float,
    PRIMARY KEY (artists, popularity, track_id)
) WITH CLUSTERING ORDER BY (popularity ASC, track_id ASC)
    AND additional_write_policy = '99p'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrate
gy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '16', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND memtable = 'default'
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND extensions = {}
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99p';

 cqlsh:music_keyspace> select count(*) from
music_by_artists_popularity;

 count
-------
 90340

(1 rows)

Warnings :
Aggregation query used without partition key

cqlsh:music_keyspace>
```

```
cqlsh:music_keyspace> CREATE TABLE music_by_track_genre
(track_genre text,track_id text, artists text, album_name text,
track_name text, popularity int, duration_ms int, explicit
boolean, danceability float, energy float,key int,loudness
float,mode int,speechiness float,acousticness
float,instrumentalness float,liveness float,valence
float,tempo float,time_signature int,PRIMARY KEY
(track_genre,track_id));
cqlsh:music_keyspace> describe music_by_track_genre

CREATE TABLE music_keyspace.music_by_track_genre (
    track_genre text,
    track_id text,
    acousticness float,
    album_name text,
    artists text,
    danceability float,
    duration_ms int,
    energy float,
    explicit boolean,
    instrumentalness float,
    key int,
    liveness float,
    loudness float,
    mode int,
    popularity int,
    speechiness float,
    tempo float,
    time_signature int,
    track_name text,
    valence float,
    PRIMARY KEY (track_genre, track_id)
) WITH CLUSTERING ORDER BY (track_id ASC)
    AND additional_write_policy = '99p'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrate
gy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '16', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND memtable = 'default'
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND extensions = {}
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99p';
cqlsh:music_keyspace>

cqlsh:music_keyspace> select count(*) from
music_by_track_genre;

 count
--------
 113421

(1 rows)

Warnings :
Aggregation query used without partition key

cqlsh:music_keyspace>
```