Διαχείριση Δεδομένων Μεγάλης Κλίμακας - Εξαμηνιαία Εργασία

Κωνσταντίνος Παπαδάκης

ΕΔΕΜΜ 03400149

konstantinospapadakis@mail.ntua.gr

Ζητούμενο 1

Φορτώνουμε τα αρχεία στο HDFS με την εντολή `hdfs dfs -put files /user/user/`, αφού πρώτα έχουμε δημιουργήσει τους φακέλους /user/ και /user/user με την εντολή `hdfs dfs -mkdir -p /user/user`.

Ζητούμενο 2

Παράγουμε parquet αρχεία τα από τα csv αρχεία και τα αποθηκεύουμε στον ίδιο φάκελο. Ο σχετικός κώδικας βρίσκεται στην ενότητα Setup του main.ipynb.

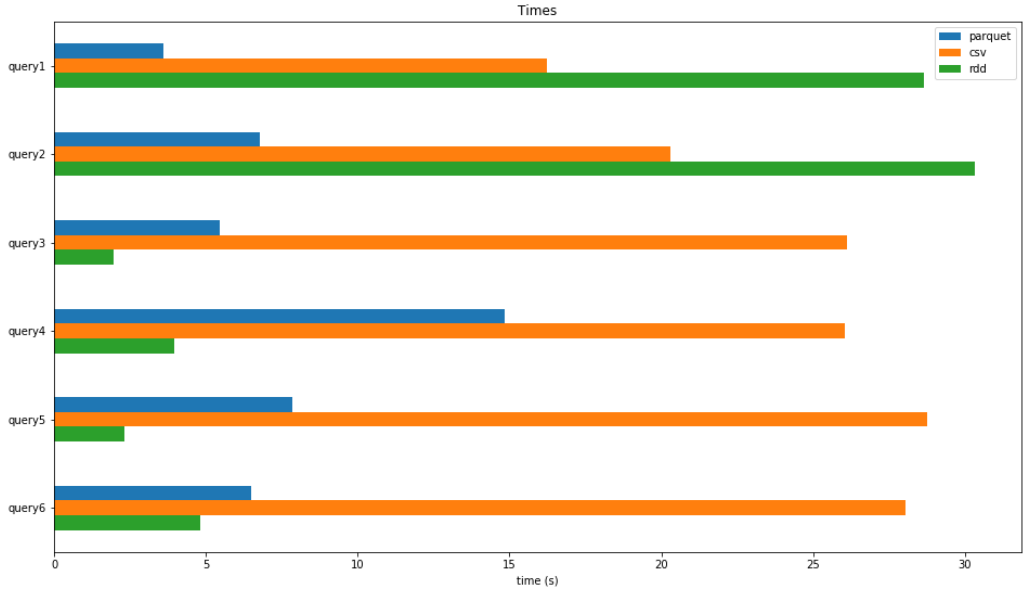
Ζητούμενο 3

Οι υλοποιήσεις των λύσεων των ερωτημάτων βρίσκονται στις ενότητες Query 1-6 του main.ipynb.

Ζητούμενο 4

Τα αποτελέσματα των χρόνων εκτελέσεων των υλοποιήσεων φαίνονται στον παρακάτω πίνακα και απεικονίζονται στην παρακάτω εικόνα.

	parquet	**csv**	**rdd**
query1	3.595705	16.230797	28.637761
query2	6.770214	20.314216	30.336148
query3	5.466290	26.108440	1.977696
query4	14.847329	26.043592	3.972164
query5	7.854828	28.762461	2.310557
query6	6.481601	28.033552	4.818343



Παρατηρούμε πως τα SQL query όταν εφαρμόζονται στα αρχεία Parquet είναι σχεδόν κατα μία τάξη μεγέθους πιο γρήγορα σε σχέση με το όταν εφαρμόζονται σε αρχεία CSV. Αυτό διότι το Parquet είναι ένα format βασισμένο σε στήλες και αυτό αρμόζει σε αναλυτικά ερωτήματα όπως τα δικά μας, ενώ από την άλλη το CSV θεωρείται από το Spark ως γραμμές αντί για στήλες.

Η υλοποίηση με RDD είναι γενικά πιο αργή, μιας και δεν γίνονται βελτιστοποιήσεις χρησιμοποιώντας γνώση για τους τύπους των αντικειμένων του RDD τα οποία είναι απλά pickleable αντικείμενα της python (σε αντίθεση με τα Dataframes που έχουν συγκεκριμένους τύπους, το Schema). Όμως το RDD πολλές φορές μας επιτρέπει να εισάγουμε λογικές βελτιστοποιήσεις οι οποίες δεν μπορούν να γίνουν αυτόματα με το SQL API και για αυτό είναι δυνατόν υλοποιήσεις με το RDD API να είναι κατά πολύ πιο γρήγορες από υλοποιήσεις με SQL.

Να σημειωθεί εδώ ότι το Parquet δεν έχει επιλογή inferSchema διότι οι οδηγίες για το σχήμα των δεδομένων είναι γραμμένες μέσα στο ίδιο το αρχείο. Το CSV από την άλλη, επειδή είναι ένα απλό αρχείο κειμένου, χρειάζεται οδηγίες για το Schema των δεδομένων, τις οποίες είτε τις παίρνει ρητά μέσω της επιλογής schema=Schema, είτε το αφήνουμε να προσπαθήσει να το ανακαλύψει αυτόματα με κάνοντας ένα πέρασμα από όλο το CSV με την επιλογή inferSchema=True. Στην υλοποίηση χρησιμοποιήθηκε ρητά το Schema.

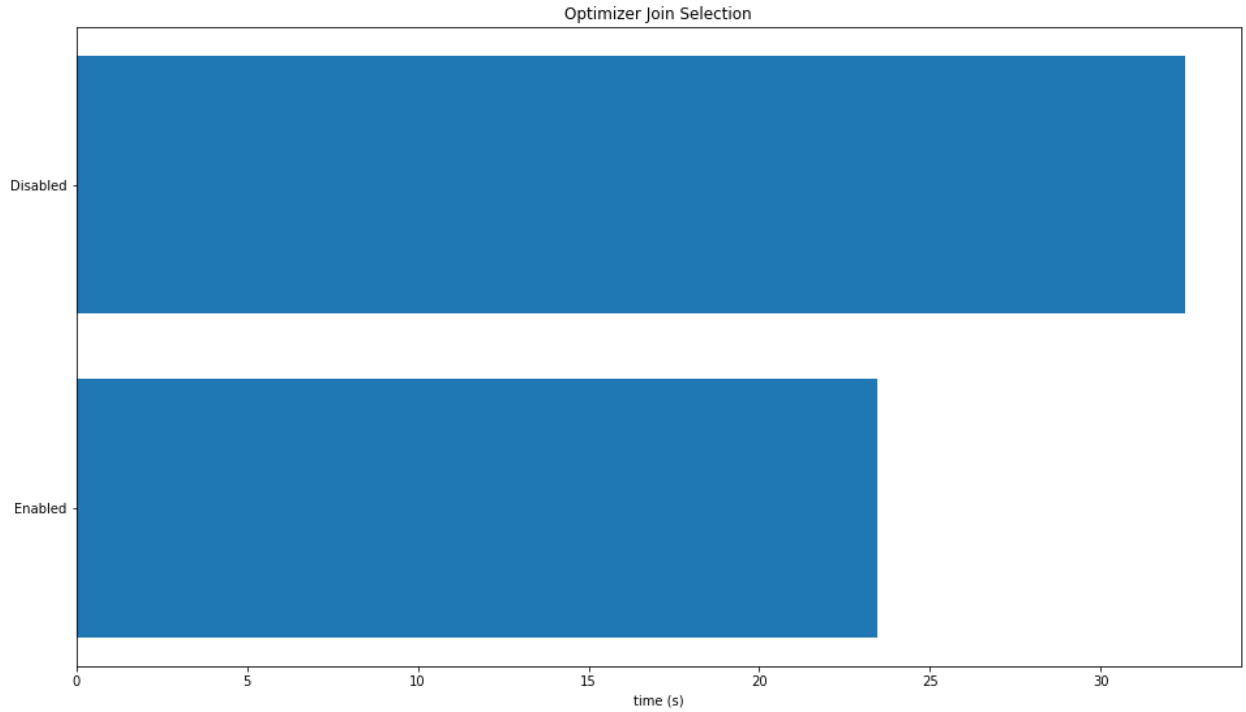
- **Query 1**: Παρατηρούμε πως το SQL API είναι πολύ πιο γρήγορο από το RDD. Αυτό πιθανόν οφείλεται διότι το ερώτημα είναι είναι πολύ απλό και συνηθισμένο και ίσως να έχει βελτιστοποιηθεί κατά πολύ. Ή ίσως να είναι προυπολογισμένα τα sums για τα partition αυτά.
- **Query 2:** Συναντάμε παρόμοια διαφορά με αυτή στο Query 1. Ίσως να οφείλεται στον ίδιο λόγο.
- **Query 3:** Εδώ βλέπουμε πως το RDD ήταν κάπως πιο γρήγορο, παρόλο που η υλοποίησή είναι ισοδύναμη με το SQL query.
- **Query 4:** Παρατηρούμε το RDD είναι πιο γρήγορο. Πιθανόν να γίνεται λιγότερο αποδοτικό join στο SQL.
- **Query 5:** Έχουμε παρόμοια αποτελέσματα με το Query 4, πιθανόν για τον ίδιο λόγο.
- **Query 6:** Οι χρόνοι εδώ ήταν περίπου ίσοι, παρόλο που στο RDD η εύρεση των streak γίνεται στην Python.

Ζητούμενο 5

Τα πλάνο φαίνεται στην ακόλουθη εικόνα

```
*****************   PHYSICAL PLAN OF JOIN EXPERIMENT WITH OPTIMIZER JOIN SELECTION (DISABLED = False)
== Physical Plan ==
*(2) Project [songId#0L, title#1, position#2, date#3, countryId#4L, chartName#5, movement#6, streams#7L, countryName#17
AS region_name#39]
+- *(2) BroadcastHashJoin [countryId#4L], [countryId#16L], Inner, BuildRight
   :- *(2) Project [songId#0L, title#1, position#2, date#3, countryId#4L, chartName#5, movement#6, streams#7L]
   :  +- *(2) Filter isnotnull(countryId#4L)
   :     +- *(2) FileScan parquet [songId#0L,title#1,position#2,date#3,countryId#4L,chartName#5,movement#6,streams#7L] B
atched: true, Format: Parquet, Location: InMemoryFileIndex[hdfs://master:9000/user/user/files/charts.parquet], Partition
Filters: [], PushedFilters: [IsNotNull(countryId)], ReadSchema: struct<songId:bigint,title:string,position:smallint,date
:date,countryId:bigint,chartName:string,m ...
   +- BroadcastExchange HashedRelationBroadcastMode(List(input[0, bigint, true]))
      +- *(1) Project [countryId#16L, countryName#17]
         +- *(1) Filter isnotnull(countryId#16L)
            +- *(1) FileScan parquet [countryId#16L,countryName#17] Batched: true, Format: Parquet, Location: InMemoryFi
leIndex[hdfs://master:9000/user/user/files/regions.parquet], PartitionFilters: [], PushedFilters: [IsNotNull(countryId)]
, ReadSchema: struct<countryId:bigint,countryName:string>
*****************   PHYSICAL PLAN OF JOIN EXPERIMENT WITH OPTIMIZER JOIN SELECTION (DISABLED = True)
== Physical Plan ==
*(5) Project [songId#49L, title#50, position#51, date#52, countryId#53L, chartName#54, movement#55, streams#56L, country
Name#66 AS region_name#88]
+- *(5) SortMergeJoin [countryId#53L], [countryId#65L], Inner
   :- *(2) Sort [countryId#53L ASC NULLS FIRST], false, 0
   :  +- Exchange hashpartitioning(countryId#53L, 200)
   :     +- *(1) Project [songId#49L, title#50, position#51, date#52, countryId#53L, chartName#54, movement#55, streams#
56L]
   :        +- *(1) Filter isnotnull(countryId#53L)
   :           +- *(1) FileScan parquet [songId#49L,title#50,position#51,date#52,countryId#53L,chartName#54,movement#55,
streams#56L] Batched: true, Format: Parquet, Location: InMemoryFileIndex[hdfs://master:9000/user/user/files/charts.parqu
et], PartitionFilters: [], PushedFilters: [IsNotNull(countryId)], ReadSchema: struct<songId:bigint,title:string,position
:smallint,date:date,countryId:bigint,chartName:string,m ...
   +- *(4) Sort [countryId#65L ASC NULLS FIRST], false, 0
      +- Exchange hashpartitioning(countryId#65L, 200)
         +- *(3) Project [countryId#65L, countryName#66]
            +- *(3) Filter isnotnull(countryId#65L)
               +- *(3) FileScan parquet [countryId#65L,countryName#66] Batched: true, Format: Parquet, Location: InMemor
yFileIndex[hdfs://master:9000/user/user/files/regions.parquet], PartitionFilters: [], PushedFilters: [IsNotNull(countryI
d)], ReadSchema: struct<countryId:bigint,countryName:string>
```

Και οι αντίστοιχοι χρόνοι φαίνονται παρακάτω



Παρατηρούμε πως όταν ο βελτιστοποιητής δεν χρησιμοποιείται τότε η διαδικασία δεν είναι αποδοτική. Για παράδειγμα βλέπουμε πως αντί να σταλεί ο regions που έχει μόνο 69 γραμμές

και 2 στήλες κοντά στον charts ώστε να γίνει BroadcastHashJoin, η στρατηγική που ακολουθείται είναι η default που που είναι η SortMergeJoin.

Περιγραφή των υλοποιήσεων

Query 1

- Filter μόνο το top200 chart και το τραγούδι shape of you
- Map x → x.streams
- Reduce με το άθροισμα

Query 2

- Filter στο position 1
- Map σε (chartName, title), 1
- ReduceByKey για τα πάρουμε τα counts
- Map σε chartName, (title, count/69). count/69 είναι το avg
- ReduceByKey όπου παίρνουμε το max ως πρός την δεύτερη θέση, δηλαδή το avg.

Query 3

- Filter στο position 1 και στο charName top200.
- Map σε (date, streams)
- ReduceByKey με το άθροισμα για να πάρουμε το συνολικό ημερήσιο πλήθος streams, ας το ονομάσουμε streamsDay.
- Map σε (year, month), (streamsDay, 1)
- ReduceByKey με vector addition, και έτσι παίρνουμε streamsYearMonth, nDaysYearMonth
- Map values στο streamsYearMonth / nDaysYearMonth για να βρούμε τον μέσο όρο για τον κάθε μήνα μία χρονιάς.

Query 4

- Αρχικά υπολογίζουμε τα counts για κάθε χώρα και τραγούδι.
- Στη συνέχεια υπολογίζουμε το maxCounts για κάθε χώρα.
- Κανουμε maxCounts join regions, γιατί είναι ακριβώς όλες οι χώρες και από τις δύο μεριές, και ονομάζουμε το αποτέλεσμα maxCountsNamed.
- Κάνουμε join τα counts με τα maxCountsNamed ως προς την χώρα
- Filter όπου count == maxCount

Query 5

- Filter στα top200
- Map σε songId, (year, streams)
- Join με chart_artist_mapping
- Map σε (year, artistId), streams
- ReduceByKey με άθροισμα και έτσι παίρνουμε (year, artistId), sumStreams
- Map σε year, (artistId, sumStreams / 69) όπου θα ονομάζουμε sumStreams / 69 avgStreams
- ReduceByKey όπου η συνάρτηση είναι το max ως προς avgStream, που μας δίνει το maxAvgStream.
- Map σε artistId, (year, maxAvgStreams).
- Join με artists
- Sort και επίστρεψε

Query 6

Να σημειωθεί εδώ πως τα ενδεικτικά αποτελέσματα δεν φαίνεται να είναι σωστά. Για παράδειγμα το “viral50,2017,21 Savage,13” βλέπουμε στο αποτέλεσμα του παρακάτω query ότι οι συνεχόμενες ημέρες στην κορυφή είναι 11 και όχι 13.

```
SELECT *
FROM charts
JOIN chart_artist_mapping
ON charts.songId = chart_artist_mapping.songId
JOIN artists
ON chart_artist_mapping.artistId = artists.artistId
WHERE countryId = 23
    AND position = 1
    AND artists.artistName = '21 Savage'
    AND year(date) = 2017
    AND chartName = 'viral50'
ORDER BY date
```

songId	title	position	date	countryId	chartName	movement	streams	songId	artistId	artistId	artistName
178991	rockstar	1	2017-10-01	23	viral50	SAME_POSITION	null	178991	401	401	21 Savage
178991	rockstar	1	2017-10-02	23	viral50	SAME_POSITION	null	178991	401	401	21 Savage
178991	rockstar	1	2017-10-03	23	viral50	SAME_POSITION	null	178991	401	401	21 Savage
178991	rockstar	1	2017-10-04	23	viral50	SAME_POSITION	null	178991	401	401	21 Savage
178991	rockstar	1	2017-10-05	23	viral50	SAME_POSITION	null	178991	401	401	21 Savage
178991	rockstar	1	2017-10-06	23	viral50	SAME_POSITION	null	178991	401	401	21 Savage
178991	rockstar	1	2017-10-07	23	viral50	SAME_POSITION	null	178991	401	401	21 Savage
178991	rockstar	1	2017-10-08	23	viral50	SAME_POSITION	null	178991	401	401	21 Savage
178991	rockstar	1	2017-10-09	23	viral50	SAME_POSITION	null	178991	401	401	21 Savage
178991	rockstar	1	2017-10-10	23	viral50	SAME_POSITION	null	178991	401	401	21 Savage
178991	rockstar	1	2017-10-11	23	viral50	SAME_POSITION	null	178991	401	401	21 Savage
178991	rockstar	1	2017-10-16	23	viral50	MOVE_UP	null	178991	401	401	21 Savage
178991	rockstar	1	2017-10-28	23	viral50	MOVE_UP	null	178991	401	401	21 Savage
178991	rockstar	1	2017-10-29	23	viral50	SAME_POSITION	null	178991	401	401	21 Savage
178991	rockstar	1	2017-11-04	23	viral50	MOVE_UP	null	178991	401	401	21 Savage
178991	rockstar	1	2017-11-05	23	viral50	SAME_POSITION	null	178991	401	401	21 Savage
178991	rockstar	1	2017-11-07	23	viral50	MOVE_UP	null	178991	401	401	21 Savage
178991	rockstar	1	2017-11-10	23	viral50	MOVE_UP	null	178991	401	401	21 Savage

Ο υπολογισμός με RDD εχει ως εξής

- Υπολογισμός streaks
    - Filter countryId = 23, position = 1
    - Map σε (chartName, year, songId), (date,)
    - ReduceByKey tuple addition με αποτέλεσμα (date1, date2, …)
    - Map τις τιμές με μία συνάρτηση που κάνει sort την ακολουθία υπολογίζει το μεγαλύτερο streak

```
def bestStreak(seq):
    seq = sorted(seq)
    res = 0
    streak = 1
    for prev, nxt in zip(seq, seq[1:]):
        if (nxt - prev).days == 1:
            streak += 1
        else:
            res = max(res, streak)
            streak = 1
    res = max(res, streak)
    return res
```

- Υπολογισμός maxStreaks με αποτέλεσμα (chartName, year), maxStreak
- Υπολογισμός τελικού αποτελέσματος
    - Map τα streaks στο (chartName, year, maxStreak), songId

- Join με maxStreaks αφού πρώτα το κάνουμε να έχει τα ίδια κλειδιά
- Map στο songId, (chartName, year, maxStreak)
- Join με το chart_artist_mapping
- Map στο artistId, (chartName, year, maxStreak)
- Join με το artists
- Sort και επίστρεψε