# ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

## ΠΡΟΧΩΡΗΜΕΝΑ ΘΕΜΑΤΑ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

Ακ. έτος 2019-2020

Εξαμηνιαία Εργασία

Η εργασία είναι **ατομική**. Για την εργασία επιλέγετε **ένα** από τα τρία θέματα.

Ακολουθεί περιγραφή των δεδομένων που θα χρησιμοποιηθούν και στα τρία θέματα.

Περιγραφή Δεδομένων

Τα δεδομένα που θα χρησιμοποιήσετε είναι πραγματικά και αφορούν σε διαδρομές taxi στην Νέα Υόρκη. Οι δοθείσες διαδρομές των taxi έγιναν από τον Ιανουάριο εώς το Ιούνιο του 2015 και υπάρχουν διαθέσιμες online στο παρακάτω link:
https://data.cityofnewyork.us/Transportation/2015-Yellow-Taxi-Trip-Data/ba8s-jw6u.

Λόγω των περιορισμένων πόρων που η κάθε ομάδα έχει στη διάθεσή της, θα επεξεργαστούμε μόνο ένα υποσύνολο μεγέθους 2 GB. Τα δεδομένα αυτά περιέχουν 13 εκατομμύρια διαδρομές, που πραγματοποιήθηκαν το Μάρτιο του 2015 και μπορείτε να τα κατεβάσετε από εδώ:
http://www.cslab.ntua.gr/courses/atds/yellow_trip_data.zip.

Στο συμπιεσμένο αρχείο που σας δίνουμε, περιλαμβάνονται δύο comma-delimited αρχεία κειμένου (.csv) που ονομάζονται: *yellow_tripdata_1m.csv* και *yellow_tripvendors_1m.csv*. Το πρώτο αρχείο περιλαμβάνει όλη την απαραίτητη πληροφορία για μια διαδρομή. Το αρχείο των TripData έχει την εξής μορφή:

*yellow_tripdata_1m.csv*

```
369367789289,2015-03-27 18:29:39,2015-03-27
 19:08:28,-73.975051879882813,40.760562896728516,-73.847900390625,40.7326850
 89111328,34.8
369367789290,2015-03-27 18:29:40,2015-03-27
 18:38:35,-73.988876342773438,40.77423095703125,-73.985160827636719,40.76343
 9178466797,11.16
```

Το πρώτο πεδίο αποτελεί το μοναδικό id μιας διαδρομής. Το δεύτερο (τρίτο) πεδίο την ημερομηνία και ώρα έναρξης (λήξης) της διαδρομής. Το τέταρτο και πέμπτο πεδίο το γεωγραφικό μήκος και πλάτος του σημείου επιβίβασης, ενώ το έκτο και έβδομο πεδίο περιλαμβάνουν το γεωγραφικό μήκος και πλάτος του σημείου αποβίβασης. Τέλος, το όγδοο πεδίο δείχνει το συνολικό κόστος της διαδρομής.

Το δεύτερο αρχείο που σας δίνεται περιέχει πληροφορία για τις εταιρίες taxi. Η μορφή του φαίνεται στο παρακάτω παράδειγμα:

*yellow_tripvendors_1m.csv*

```
369367789289,1
369367789290,2
```

Το πρώτο πεδίο αποτελεί το μοναδικό id μιας διαδρομής και το δεύτερο πεδίο το μοναδικό αναγνωριστικό μιας εταιρείας taxi (vendor).

## Θέμα 1ο: Υλοποίηση SQL ερωτημάτων για αναλυτική επεξεργασία δεδομένων

Όπως αναφέρθηκε τα δεδομένα σας δίνονται σε μορφή απλού κειμένου (csv). Παρόλα αυτά, είναι γνωστό ότι ο υπολογισμός ερωτημάτων αναλυτικής επεξεργασίας απευθείας πάνω σε αρχεία csv δεν είναι αποδοτικός. Για να βελτιστοποιηθεί η πρόσβαση των δεδομένων, παραδοσιακά οι βάσεις δεδομένων φορτώνουν τα δεδομένα σε ειδικά σχεδιασμένα binary formats.
Παρ'ότι το Spark δεν είναι μια τυπική βάση δεδομένων, αλλά ένα σύστημα κατανεμημένης επεξεργασίας, για λόγους απόδοσης, υποστηρίζει κι αυτό μια παρόμοια λογική. Αντί να τρέξουμε τα ερωτήματά μας απευθείας πάνω στα csv αρχεία, μπορούμε να μετατρέψουμε πρώτα το dataset σε μια ειδική μορφή που:

- Έχει μικρότερο αποτύπωμα στη μνήμη και στον δίσκο και άρα βελτιστοποιεί το I/O (input/output) μειώνοντας τον χρόνο εκτέλεσης.
- Διατηρεί επιπλέον πληροφορία, όπως στατιστικά πάνω στο dataset, τα οποία βοηθούν στην πιο αποτελεσματική επεξεργασία του. Για παράδειγμα, αν ψάχνω σε ένα σύνολο δεδομένων τις τιμές που είναι μεγαλύτερες από 100 και σε κάθε block του dataset έχω πληροφορία για το ποια είναι η min και ποια η max τιμή, τότε μπορώ να παρακάμψω την επεξεργασία των blocks με max τιμή < 100 γλιτώνοντας έτσι χρόνο επεξεργασίας.

Το ειδικό format που χρησιμοποιούμε για να επιτύχουμε τα παραπάνω είναι το Apache Parquet. Όταν φορτώνουμε έναν πίνακα σε Parquet, αυτός μετατρέπεται κι αποθηκεύεται σε ένα columnar format που βελτιστοποιεί το I/O και τη χρήση της μνήμης κι έχει τα χαρακτηριστικά που αναφέραμε. Περισσότερες πληροφορίες σχετικά με το Parquet μπορείτε να βρείτε εδώ: https://parquet.apache.org/.
Από άποψη κώδικα, η μετατροπή ενός dataset σε Parquet είναι ιδιαίτερα απλή. Παραδείγματα και πληροφορίες για το πώς διαβάζω και γράφω Parquet αρχεία μπορείτε να βρείτε εδώ: https://spark.apache.org/docs/latest/sql-data-sources-parquet.html

Το πρόβλημα που καλείστε να αντιμετωπίσετε είναι ο υπολογισμός των ερωτημάτων του Πίνακα 1 με δύο διαφορετικούς τρόπους:

- Γράφοντας MapReduce κώδικα χρησιμοποιώντας το RDD API του Spark
- Χρησιμοποιώντας SparkSQL και το DataFrame API

Πιο συγκεκριμένα, θα πρέπει να κάνετε τα εξής:

1. Φορτώστε τα csv αρχεία που σας δίνονται στο HDFS.
2. Υλοποιήστε και τρέξτε τα ερωτήματα του Πίνακα 1 με χρήση:
    a. MapReduce κώδικα. Η υλοποίηση θα πρέπει να τρέξει απευθείας πάνω στα αρχεία κειμένου.
    b. SparkSQL. Φορτώστε τα αρχεία κειμένου σε Dataframes και εκτελέστε τα ερωτήματα με χρήση της SparkSQL.
3. Μετατρέψτε τα αρχεία κειμένου σε αρχεία Parquet. Στη συνέχεια φορτώστε τα Parquet αρχεία ως Dataframes και εκτελέστε το υποερώτημα 2b. Πόσος χρόνος χρειάζεται για τη μετατροπή των αρχείων;
4. Για κάθε ερώτημα του Πίνακα 1, συγκρίνετε και φτιάξτε ένα διάγραμμα με τον χρόνο εκτέλεσης των παραπάνω περιπτώσεων, δηλαδή:
    a. MR πάνω σε csv αρχεία.
    b. SQL πάνω σε csv αρχεία.
    c. SQL πάνω σε Parquet αρχεία. (Διευκρίνιση: Τα SQL ερωτήματα θα πρέπει να τρέξουν σε Dataframes που έχουν δημιουργηθεί είτε από αρχεία κειμένου είτε από αρχεία Parquet).

    Σχολιάστε τα αποτελέσματά σας.

Πίνακας 1

| ID | Query |
|---|---|
| Q1 | Ποια είναι η μέση διάρκεια διαδρομής (σε λεπτά) ανά ώρα έναρξης της διαδρομής; Ταξινομείστε το αποτέλεσμα με βάση την ώρα έναρξης σε αύξουσα σειρά.<br>Ενδεικτικά αποτελέσματα:<br><br>`HourOfDay AverageTripDuration`<br>`00 13.035268962938562`<br>`01 15.24545454540119` |
| Q2 | Ποιες είναι οι 5 πιο γρήγορες κούρσες που έγιναν μετά τις 10 Μαρτίου και σε ποιους vendors ανήκουν; |

Σημειώσεις-Υποδείξεις

1. Κάθε γραμμή του αρχείου που διαβάζουμε με το RDD API φορτώνεται στη μνήμη ως string. Αφού εξάγουμε τις επιθυμητές στήλες από τη γραμμή, για να κάνουμε πράξεις με

κάποιες στήλες θα πρέπει οι τιμές να μετατραπούν από string στον κατάλληλο τύπο πρώτα. Τις ημερομηνίες π.χ. μπορούμε να τις μετατρέψουμε κατάλληλα χρησιμοποιώντας τη μορφή '%Y-%m-%d %H:%M:%S'.

2. Για το Q2, η ταχύτητα μιας κούρσας ορίζεται ως η απόσταση που διανύθηκε προς τον χρόνο που χρειάστηκε.
3. Υπολογισμός απόστασης (Haversine[1]). Αν φ είναι το γεωγραφικό πλάτος και λ το γεωγραφικό μήκος, τότε η απόσταση δύο σημείων δίνεται από τους τύπους:
   $a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$
   $c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{(1-a)})$
   $d = R \cdot c$, όπου R είναι η ακτίνα της Γης (6371m)

## Θέμα 2ο: Υλοποίηση και μελέτη απόδοσης αλγορίθμων συνένωσης για αναλυτική επεξεργασία δεδομένων

Στο δεύτερο θέμα καλείστε να μελετήσετε και να αξιολογήσετε τις διαφορετικές υλοποιήσεις που υπάρχουν στο περιβάλλον Map-Reduce του Spark για τη συνένωση (join) δεδομένων και συγκεκριμένα το repartition join (aka Reduce-Side join) (Παράγραφος 3.1 και ψευδοκώδικας Α.1 της δημοσίευσης) και το broadcast join (aka Map-Side join) (Παράγραφος 3.2 και ψευδοκώδικας Α.4) όπως έχουν περιγραφεί στην δημοσίευση “A Comparison of Join Algorithms for Log Processing in MapReduce”, Blanas et al[2], in Sigmod 2010. Το broadcast join θεωρείται πιο αποδοτικό σε περίπτωση join ενός μεγάλου fact[3] table και ενός σχετικά μικρότερου dimension table[4].

Ζητούμενα:

1. **Υλοποιήστε** τόσο τον αλγόριθμο broadcast οσο και τον repartition join σύμφωνα με το παραπάνω paper, χρησιμοποιώντας το RDD API του Spark.
2. Απομονώστε τις 25 πρώτες γραμμές από το αρχείο με τις εταιρείες ταξί. Φορτώστε το υποσύνολο αυτό και το csv των διαδρομών στο HDFS.
3. Εκτελέστε ανάμεσα στα δύο αρχείο που φορτώσατε στο HDFS τα 2 είδη ερωτημάτων συνένωσης που υλοποιήσατε. Μετρήστε στην κάθε περίπτωση το χρόνο εκτέλεσης του ερωτήματος και σχολιάστε τα αποτελέσματα σύμφωνα με τα όσα αναφέρονται στο παραπάνω paper.

Το SparkSQL έχει υλοποιημένα και τα δύο είδη ερωτημάτων συνένωσης στο DataFrame API. Συγκεκριμένα, με βάση τη δομή των δεδομένων και των υπολογισμών που θέλουμε καθώς και τις ρυθμίσεις του χρήστη, πραγματοποιεί από μόνο του κάποιες βελτιστοποιήσεις στην εκτέλεση του ερωτήματος χρησιμοποιώντας έναν βελτιστοποιητή ερωτημάτων (query optimizer), κάτι που όλες οι βάσεις δεδομένων έχουν. Μια τέτοια βελτιστοποίηση είναι ότι επιλέγει αυτόματα την

[1] https://en.wikipedia.org/wiki/Haversine_formula
[2] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.644.9902&rep=rep1&type=pdf
[3] https://en.wikipedia.org/wiki/Fact_table
[4] https://en.wikipedia.org/wiki/Dimension_(data_warehouse)

υλοποίηση που θα χρησιμοποιήσει για ένα ερώτημα join λαμβάνοντας υπόψη το μέγεθος των δεδομένων και πολλές φορές αλλάζει και την σειρά ορισμένων τελεστών προσπαθώντας να μειώσει τον συνολικό χρόνο εκτέλεσης του ερωτήματος. Αν ο ένας πίνακας είναι αρκετά μικρός (με βάση ένα όριο που ρυθμίζει ο χρήστης) θα χρησιμοποιήσει το broadcast join, αλλιώς θα κάνει ένα repartition join. Περισσότερες πληροφορίες για τις ρυθμίσεις βελτιστοποίησης του SparkSQL υπάρχουν εδώ: https://spark.apache.org/docs/latest/sql-performance-tuning.html.
Είναι σημαντικό να τονίσουμε ότι ο βελτιστοποιητής μπορεί να χρησιμοποιήσει τις απαραίτητες πληροφορίες για τα δεδομένα, όταν αυτά είναι αποθηκευμένα σε κατάλληλα formats, όπως π.χ. το Apache Parquet file format. Το συγκεκριμένο format:

- Έχει μικρότερο αποτύπωμα στη μνήμη και στον δίσκο και άρα βελτιστοποιεί το I/O (input/output) μειώνοντας τον χρόνο εκτέλεσης.
- Διατηρεί επιπλέον πληροφορία, όπως στατιστικά πάνω στο dataset, τα οποία βοηθούν στην πιο αποτελεσματική επεξεργασία του και καθιστούν πιο αποτελεσματική λειτουργία τη λειτουργία ενός optimizer.

Περισσότερες πληροφορίες για το συγκεκριμένο file format θα βρείτε εδω:
https://parquet.apache.org/.

Στη συνέχεια της εργασίας καλείστε να εκτελέσετε τα παρακάτω:

4. Μετατρέψτε τα αρχεία που φορτώσατε στο HDFS σε μορφή parquet. Η διαδικασία αυτή είναι αρκετά απλή με τη βοήθεια του Spark και περιγράφεται στον ακόλουθο σύνδεσμο. https://spark.apache.org/docs/latest/sql-data-sources-parquet.html
5. Εκτελέσετε με SparkSQL ένα join πάνω στα 2 parquet αρχεία και πάρτε πίσω όλα τα δεδομένα. Εντοπίστε ποια υλοποίηση join χρησιμοποίησε το Spark. Γιατί επιλέχθηκε η συγκεκριμένη υλοποίηση;
   *Υπόδειξη*: Μπορείτε να χρησιμοποιήσετε το ‘explain’ statement της SQL για να δείτε τις λεπτομέρειες του πλάνου εκτέλεσης. Αιτιολογήστε την απάντησή σας με βάση το πλάνο εκτέλεσης του ερωτήματος και τις προκαθορισμένες ρυθμίσεις του Spark.
6. Ρυθμίστε κατάλληλα το Spark χρησιμοποιώντας τις ρυθμίσεις του βελτιστοποιητή ώστε να μην επιλέγει την υλοποίηση join του προηγούμενου ερωτήματος. Σε πόσο χρόνο εκτελείται τώρα το ερώτημα και ποια υλοποίηση join χρησιμοποίησε το Spark αυτή τη φορά; Συγκρίνετε τον χρόνο εκτέλεσης των δύο υποερωτημάτων. Προκύπτουν τα ίδια συμπεράσματα με τη δική σας υλοποίηση;

## Θέμα 3ο: Machine Learning - Ομαδοποίηση δεδομένων με εκτέλεση του k-means αλγόριθμου

Χρησιμοποιώντας τα δεδομένα που περιγράψαμε, στο τρίτο θέμα θέλουμε να βρούμε τις κεντρικές συντεταγμένες των top 5 περιοχών επιβίβασης πελατών. Για το σκοπό αυτό ζητείται **να υλοποιηθεί** ο αλγόριθμος ομαδοποίησης k-means (https://en.wikipedia.org/wiki/K-means_clustering), τον οποίο θα χρησιμοποιήσουμε για να

ομαδοποιήσουμε τα σημεία επιβίβασης σε k=5 περιοχές (clusters) και να βρούμε το κέντρο των σημείων κάθε περιοχής.

Ο αλγόριθμος είναι επαναληπτικός και θεωρώντας 5 αρχικά κέντρα με προκαθορισμένες συντεταγμένες, γίνονται δύο βήματα σε κάθε επανάληψη:

1. *Ανάθεση σημείων σε περιοχή*: βρίσκουμε και αντιστοιχούμε κάθε σημείο επιβίβασης στο κοντινότερο από τα 5 κέντρα.
2. *Ανανέωση κέντρων περιοχών*: υπολογίζουμε τις συντεταγμένες των 5 νέων κέντρων ως τον μέσο όρο των συντεταγμένων των σημείων που ανατέθηκαν σε κάθε περιοχή.

Ο αλγόριθμος συγκλίνει όταν τα κέντρα δεν αλλάζουν πλέον από επανάληψη σε επανάληψη.

Φορτώστε αρχικά τα csv αρχεία που σας δίνονται στο HDFS. Για την υλοποίησή σας θεωρείστε ως αρχικά κέντρα τις συντεταγμένες επιβίβασης από τις 5 πρώτες γραμμές των δεδομένων μας και ότι χρειάζονται 3 μόνο επαναλήψεις του αλγορίθμου ώστε να επιτευχθεί σύγκλιση. Ακολουθεί ψευδοκώδικας της υλοποίησης:

```
points = read_data()
k = 5
MAX_ITERATIONS = 3

# Initialize centroids
centroids = get_k_first_points(points, k)
iterations = 1
while not (iterations > MAX_ITERATIONS) {
    iterations += 1

    # For each point in the dataset, chose the closest centroid.
    # Make that centroid's index the point's label.
    points_and_labels = get_labels(points, centroids)

    # Each new centroid is the mean of the points that have the
    # same centroid's label.
    new_centroids = get_new_centroids(points_and_labels)
    centroids = new_centroids
}
print(centroids)
```

Αποθηκεύσετε σε ένα αρχείο στο HDFS το αποτέλεσμα που θα έχει ενδεικτικά την εξής πληροφορία:

```
Centroid   Coordinates
```

```
1           [-74.50469347,40.74525183]
2           [-74.51469347,40.75525183]
3           [-74.52469347,40.76525183]
4           [-74.53469347,40.77525183]
5           [-74.54469347,40.78525183]
```

Σημειώσεις:

1. Υπολογισμός απόστασης (Haversine[5]). Αν φ είναι το γεωγραφικό πλάτος και λ το γεωγραφικό μήκος, τότε η απόσταση δύο σημείων δίνεται από τους τύπους:
   $a = \sin^2(\Delta\varphi/2) + \cos\varphi1 \cdot \cos\varphi2 \cdot \sin^2(\Delta\lambda/2)$
   $c = 2 \cdot \text{atan2}( \sqrt{a}, \sqrt{(1-a)} )$
   $d = R \cdot c$, όπου R είναι η ακτίνα της Γης (6371m)

## Παραδοτέα

- Η προθεσμία υποβολής της άσκησης είναι την Δευτέρα 3 Φεβρουαρίου 2020 23:59.
- Η άσκηση που θα επιλέξετε θα υλοποιηθεί **ατομικά**.
- Η παράδοση θα γίνει στο mycourses site.
- Η παράδοση θα αποτελείται από:
    - Μια σύντομη αναφορά όπου θα περιγράφετε την μεθοδολογία που ακολουθήσατε (όχι κώδικας εδώ).
    - Ψευδοκώδικας για τα προγράμματα Map/Reduce που χρησιμοποιήσατε για κάθε κομμάτι της άσκησης. Ο ψευδοκώδικας θα δείχνει εποπτικά τα key/values που παίρνει η συνάρτηση map, την επεξεργασία που τους κάνει, τα key/values που κάνει emit στην συνάρτηση reduce, και την επεξεργασία που κάνει η reduce (σαν τον ψευδοκώδικα του wordcount).
    - Link στο hdfs site όπου έχετε βάλει τα datasets.
    - Ένα zip file με τον κώδικα.
    - Ένα zip file με τα τελικά αποτελέσματα.
    - Ένα zip file με τα log-files των εργασιών MapReduce από τις οποίες βγήκαν τα αποτελέσματα.

[5] https://en.wikipedia.org/wiki/Haversine_formula