ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ
ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

ΔΙΑΧΕΙΡΙΣΗ ΔΕΔΟΜΕΝΩΝ ΜΕΓΑΛΗΣ ΚΛΙΜΑΚΑΣ

Ακ. έτος 2018-2019

Ν. Κοζύρης, Γ. Κωνσταντίνου, Γιάγκος Μυτιλήνης, Ευδοκία Κασσέλα

Εξαμηνιαία Εργασία

Σκοπός της φετινής εργασίας του μαθήματος είναι η ανάπτυξη δεξιοτήτων αναλυτικής επεξεργασίας καθώς και η εξοικείωση με την πλατφόρμα Apache Spark (https://spark.apache.org/). Τα θέματα, τα οποία καλείστε να υλοποιήσετε, ανταποκρίνονται σε πραγματικές προκλήσεις που αντιμετωπίζει ένας data engineer και στοχεύουν τόσο στην ανάπτυξη αλγορίθμων εξαγωγής γνώσης από ένα σύνολο δεδομένων, όσο και στην πειραματική διερεύνηση της απόδοσης ενός συστήματος κάτω από διαφορετικές παραμέτρους.

Δεδομένα

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 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).

Μορφή Δεδομένων

Όπως αναφέρθηκε τα δεδομένα σας δίνονται σε μορφή απλού κειμένου (txt). Παρόλα αυτά, είναι γνωστό ότι ο υπολογισμός ερωτημάτων αναλυτικής επεξεργασίας απευθείας πάνω σε αρχεία txt 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 txt αρχεία, μπορούμε να μετατρέψουμε πρώτα το dataset σε μια ειδική μορφή που:

1. Έχει μικρότερο αποτύπωμα στη μνήμη και στον δίσκο και άρα βελτιστοποιεί το I/O (input/output) μειώνοντας τον χρόνο εκτέλεσης.
2. 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

Θέμα 1ο: Υλοποίηση SQL ερωτημάτων για αναλυτική επεξεργασία δεδομένων (80 μονάδες)

Το πρώτο 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 txt αρχεία που σας δίνονται στο HDFS. (10 μονάδες)
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. (30 μονάδες)
    b. SparkSQL. Φορτώστε τα αρχεία κειμένου σε Dataframes και εκτελέστε τα ερωτήματα με χρήση της SparkSQL. (20 μονάδες)
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; (10 μονάδες)
4. Για κάθε ερώτημα του Πίνακα 1, συγκρίνετε και φτιάξτε ένα διάγραμμα με τον χρόνο εκτέλεσης των παραπάνω περιπτώσεων, δηλαδή:
    a. MR πάνω σε txt αρχεία.
    b. SQL πάνω σε txt 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. (10 μονάδες)

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. Ενδεικτικά αποτελέσματα: `HourOfDay  AverageTripDuration`

	00 13.035268962938562 01 15.24545454540119
Q2	Ποιο είναι το μέγιστο ποσό που πληρώθηκε σε μία διαδρομή για κάθε εταιρεία ταξί (vendor);
Q3	Ποιες είναι οι 5 πιο γρήγορες κούρσες που έγιναν μετά τις 10 Μαρτίου και σε ποιούς 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. 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)
3. Για το Q3, η ταχύτητα μιας κούρσας ορίζεται ως η απόσταση που διανύθηκε προς τον χρόνο που χρειάστηκε.

Θέμα 2ο: Μελέτη απόδοσης αλγορίθμων συνένωσης στο Apache Spark (20 μονάδες)

Στο δεύτερο πρόβλημα 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].
Το SparkSQL έχει δημιουργηθεί ώστε να λαμβάνει υπόψη τη δομή των δεδομένων και των υπολογισμών που θέλουμε να κάνουμε καθώς και τις ρυθμίσεις του χρήστη και να πραγματοποιεί από μόνο του κάποιες βελτιστοποιήσεις στην εκτέλεση του ερωτήματος

[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)

χρησιμοποιώντας έναν βελτιστοποιητή ερωτημάτων (query optimizer), κάτι που όλες οι βάσεις δεδομένων έχουν. Μια τέτοια βελτιστοποίηση είναι ότι επιλέγει αυτόματα την 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.
Ζητούμενα του θέματος είναι τα εξής:

1. Εκτελέσετε με SparkSQL ένα join πάνω στα 2 parquet αρχεία που κατασκευάσατε στο προηγούμενο θέμα και πάρτε πίσω όλα τα δεδομένα. Διαλέξτε μόνο τις πρώτες 100 γραμμές από το αρχείο με τις εταιρίες taxi (πριν εκτελέσετε το join). Εντοπίστε ποια υλοποίηση join χρησιμοποίησε το Spark. Γιατί επιλέχθηκε η συγκεκριμένη υλοποίηση; (10 μονάδες)
   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 και τις προκαθορισμένες ρυθμίσεις του Spark.
2. 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 σε σύγκριση με το υποερώτημα 1; Εντοπίστε ποια υλοποίηση join χρησιμοποίησε τώρα το Spark και αιτιολογήστε πως εξηγείται η διαφορά στο χρόνο εκτέλεσης. (10 μονάδες)

Παραδοτέα

- Η άσκηση θα υλοποιηθεί είτε ατομικά, είτε σε ομάδες 2 ατόμων.
- Η παράδοση θα γίνει στο mycourses site.
- Η προθεσμία παράδοσης είναι την Κυριακή 8 Σεπτεμβρίου 2019 23:59.
- 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.