ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΕΜΠ

Φιλίππου Μιχαήλ
el15756

Προηγμένα Θέματα Βάσεων Δεδομένων
Εισαγωγή στο Map Reduce

Εισαγωγή

Στην εργασία αυτή θα χρησιμοποιήσουμε αλγορίθμους MapReduce για να κάνουμε αποθήκευση και ανάλυση σε έναν αριθμό από σετ δεδομένων . Για την υλοποίηση της εργασίας αυτής έχουμε στην διάθεση μας ένα cluster στον ωκεανό το οποίο αποτελείται από 2 υπολογιστές καθένας από τους οποίους διαθέτει 2 πυρήνες cpu , 2gb Ram και 30gb μνήμη . Επιπρόσθετα , για την υλοποίηση της εργασίας αυτής θα χρησιμοποιήσουμε τόσο το Hadoop distributed filesystem όσο και το Spark RDD API. Πιο συγκεκριμένα θα χρησιμοποιήσουμε το hdfs για να αποθηκεύσουμε το input και output του κώδικα μας σε ένα distributed σύστημα , ενώ θα χρησιμοποιήσουμε το Spark RDD API για να προγραμματίσουμε τα MapReduce που θα εκτελεί ο κώδικας μας πάνω στα δεδομένα εισόδου.

Θέμα 1ο: Υλοποίηση SQL ερωτημάτων για αναλυτική επεξεργασία δεδομένων

Αρχικά χωρίσαμε την άσκηση σε 2 αρχεία (ένα για κάθε ερώτημα) με 3 μέρη το καθένα ανάλογα την υλοποίηση (RDD, SQL με csv και SQL με parquet).
Ξεκινώντας με το Q1, στην RDD υλοποίηση, ξεκινάμε φορτώνοντας το csv αρχείο στο hdfs για να μπορεί να γίνει η επεξεργασία με το spark.Έπειτα, χρησιμοποιώντας την συνάρτηση map() του spark, εφαρμόζουμε την συνάρτηση getDuration() η οποία δέχεται ως όρισμα μία γραμμή από το csv αρχείο και επιστρέφει την διάρκεια μιας κούρσας σαν τιμή (value) μιας tuple με κλειδί (key) την ώρα έναρξης στην μορφή (hh).Στην συνέχεια κάνει μια ομαδοποίηση (groupBy) τις τιμές των κλειδιών και και υπολογίζουμε την average χρονική διάρκεια που συναντάμε σε κούρσες που ξεκινάνε εκείνη την ώρα.Στο επόμενο μέρος υλοποιούμε ένα sql query με την χρήση της sparksql πάνω στα δεδομένα.Στο τρίτο και τελευταίο μέρος δημιουργούμε ένα .parquet αρχείο στο οποίο εφαρμόζουμε τον ίδιο sql κώδικα.
Το Q2, στην RDD υλοποίηση, ξεκινάμε με το να φορτώσουμε και τα 2 csv files στο hdfs.Έπειτα χρησιμοποιώντας την filter() κάνουμε ένα ‘ξεσκόνισμα’ στα δεδομένα για να διώξουμε όσο μπορούμε outliers (συναρτήσεις validDate() και isNewYork()). Με την map() φτιάχνουμε tuples με στοιχεία της μορφής (id,velocity) όπου το velocity είναι σε km/h.Τέλος ταξινομούμε τα δεδομένα και επιστρέφουμε μία λίστα με τις 5 ταχύτερες διαδρομές.Αφού βρήκαμε τις ταχύτερες 5, στο δεύτερο dataset που περιέχει το id της κούρσας και τον vendor, βρίσκουμε με την χρήση της filter() τις 5 διαδρομές που είχαμε υπολογίσει προηγουμένως σε ποιον vendor ανήκουν και το επιστρέφουμε.Τέλος κάνουμε join() τα 2 αυτά datasets αφού τα έχουμε μετασχηματήσει κατάλληλα καθώς το 1ο dataset είναι σε μορφή rdd ενώ το 2ο σε λίστα.Επιστρέφουμε τις ταχύτητες και τον vendor στον οποίο ανήκουν.Όμοια υλοποιούμε τα SQL ερωτηματα με τις εκάστοτε τροποποιήσεις όπως στο προηγούμενο αρχείο.

Αναλυτικότερα:

Q1:

1. map(lambda line:(getDuration(line)):
   Επιστρέφει ένα ζεύγος (‘hh’,’duration’).
2. aggregateByKey((0,0), lambda a,b: (a[0] + b, a[1] +1),lambda a,b: (a[0] + b[0], a[1] + b[1]))
   Ομαδοποιει τις τιμές και συνενώνει τα values των duration.
3. mapValues(lambda v: v[0]/v[1])
   Επιστρέφει τον λόγο σε κάθε value του συνόλου.
4. sortBy(lambda a: a[0]).collect()
   Ταξινομεί τα στοιχεία του αρχείου με βάση το πρώτο στοιχείο του (‘hh’,’duration’).

Q2:

- Στο dataset με τις διαδρομές:
  1) Εφαρμόζουμε την filter(lambda line : validDate(line)) με όρισμα μια συνάρτηση validDate() ώστε να επιβεβαιώσουμε ότι είναι στις επιθυμητές ημερομηνίες.
  2) Εφαρμόζουμε τηνfilter(lambda line : isNewYork(line)) με όρισμα μια συνάρτηση validDate() ώστε να επιβεβαιώσουμε ότι είναι στις επιθυμητές ημερομηνίες.
  3) Εφαρμόζουμε την map(lambda line : getVelocity(line)) ώστε να έχουμε τις επιθυμητές ταχύτητες στην μορφή (id,velocity) και velocity (km/h).Η getVelocity() κάνει και μια μικρή εκκαθάριση σε outliers.
  4) Εφαρμόζουμε την top(5, key=lambda x: x[1]) ή οποία επιστρέφει τα 5 μεγαλύτερα στοιχεία τα οποία τα ταξινομεί με βάση το δεύτερο όρισμα της tuple.

- Στο dataset με του παρόχους:
  1) Εφαρμόζουμε την map(lambda s: s.split(",")).map(lambda line: (line[0],line[1])) ώστε να έχουμε tuples της μορφής (‘id’,’vendor’)
  2) Εφαρμόζουμε την .filter(lambda line : isTopFive(line,df1)) ώστε να κρατήσουμε τους 5 id με τους οποίους βρηκαμε πάνω.

Κάνω join() και επιστρέφω τα κατάλληλα πεδία.

Αποτελέσματα:

Όπως είναι αναμενόμενο παρατηρούμε μια σημαντική διαφορά στις ταχύτητες που χρειάζονται. Το RDD είναι σημαντικά πιο αργό (τουλάχιστον 3 φορες πιο αργό) σε σχέση με το sql csv.Επίσεις παρατηρούμε ότι ακόμα και για τα λίγα δεδομένα που είχαμε (τάξεως μερικών GB) υπάρχει αισθητή διαφορά μεταξύ parquet και csv.
Ανάμεσα στα 2 ερωτήματα υπάρχει επίσεις διαφορά στον χρόνο που απαιτείται καθώς θα χαρακτηριζότανι πιο απαιτητικό (περισσοτερες ενεργειες, περισσότερα δεδομένα).