Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Προχωρημένα Θέματα Βάσεων Δεδομένων (Ροή Λ) – Χειμερινό Εξάμηνο 2019-2020
Κλεάνθης Αβραμίδης – 03115117 Ημερομηνία: 10.03.2020

MapReduce for k-means Clustering

1 Εισαγωγή

Η παρούσα εργασία έχει στόχο την υλοποίηση ενός αλγορίθμου κατά MapReduce σε ένα ειδικά διαμορφωμένο Spark cluster. Για την υλοποίησή της αναπτύχθηκε ένα remote περιβάλλον master-worker μέσω της υπηρεσίας Virtual Machines του okeanos knossos, όπου εγκαταστάθηκαν το κατανεμημένο filesystem HDFS για την υποδοχή των δεδομένων και το open source περιβάλλον επεξεργασίας δεδομένων Spark, το οποίο περιλαμβάνει υλοποίηση του προγραμματιστικού μοντέλου MapReduce. Ως γλώσσα υλοποίησης επιλέχθηκε η Python3 και το αντίστοιχο πακέτο pyspark. Το script που υλοποιεί τα ζητούμενα εκτελείται μέσω της εντολής spark submit, αφού έχουν εκκινήσει το HDFS και το Spark cluster, ενώ τα αποτελέσματα γράφονται σε .txt.

Τα δεδομένα που χρησιμοποιήθηκαν στην εργασία προέρχονται από ένα dataset με 13.000.000 διαδρομές taxi στη Νέα Υόρκη την περίοδο Ιανουαρίου-Ιουνίου 2015 [1]. Συγκεκριμένα ασχοληθήκαμε με ένα υποσύνολο 2GB του συνόλου δεδομένων που αφορά τον μήνα Μάρτιο και περιλαμβάνει 2 αρχεία:

- **yellow_tripdata_1m.csv**: (route_id, timestamp_start, timestamp_end, lng_start, lat_start, lng_end, lat_end, cost)
- **yellow_tripvendors_1m.csv**: (route_id, vendor)

Επιλέξαμε να εργαστούμε πάνω στο 3ο θέμα που αφορά την εύρεση των κεντρικών συντεταγμένων των 5 συνηθέστερων περιοχών επιβίβασης μέσω του αλγορίθμου ομαδοποίησης k-means. Συνεπώς μας είναι χρήσιμο μόνο το πρώτο αρχείο το οποίο και φορτώνουμε στο HDFS μέσω της εντολής **`hadoop fs -put yellow_tripdata_1m.csv hdfs://master:9000/`**.

Παρακάτω επεξηγείται ο αλγόριθμος k-means καθώς και ο κώδικας που τον υλοποιεί. Στη συνέχεια δίνεται ψευδοκώδικας των εργασιών MapReduce που υλοποιούνται ενώ τέλος παρατίθενται τα αποτελέσματα της εκτέλεσης του προγράμματος. Εκτός της παρούσας αναφοράς, στο φάκελο των παραδοτέων περιέχεται το εκτελέσιμο script, το csv αρχείο των αποτελεσμάτων, δύο txt αρχεία με τα logs των εργασιών MapReduce καθώς και ένα αρχείο με τα links των remote servers για HDFS και Spark.

2 Αλγόριθμος

Ο αλγόριθμος k-means συγκαταλέγεται στους αλγορίθμους μηχανικής μάθησης χωρίς επίβλεψη (unsupervised learning) που χρησιμοποιείται για την ομαδοποίηση άγνωστων δεδομένων (clustering). Ο αλγόριθμος είναι επαναληπτικός και αρχικοποιώντας k τον αριθμό τυχαία κέντρα (centroids) ακολουθεί δύο κύρια στάδια:

- Αντιστοιχεί κάθε δεδομένο του χώρου στο κοντινότερό του κέντρο, σύμφωνα με μια μετρική απόστασης.
- Ανανεώνει τη θέση κάθε κέντρου υπολογίζοντας το μέσο όρο των τιμών των δεδομένων του.

Η διαδικασία αυτή επαναλαμβάνεται μέχρις ότου η θέση των κέντρων να συγκλίνει και δεν αλλάζει ουσιωδώς ανά επανάληψη. Στο συγκεκριμένο πρόβλημα μας ζητείται η εύρεση 5 κέντρων (k=5) τρέχοντας τον αλγόριθμο για 3 επαναλήψεις. Τα δεδομένα μας θα είναι οι συντεταγμένες (lng,lat) των περιοχών επιβίβασης και τα αποτελέσματα θα αποθηκεύονται σε .csv στο HDFS.

Προχωράμε τώρα στην αναλυτική επεξήγηση του κώδικα, που ούτως ή άλλως συνοδεύεται με εκτενή σχολιασμό στο script. Αρχικά ορίζουμε τα packages που θα χρησιμοποιήσουμε και αρχικοποιούμε το Spark Session στο οποίο θα δουλέψουμε, προκειμένου να έχουμε πρόσβαση στις λειτουργίες του Spark και στο μοντέλο MapReduce. Στη συνέχεια ορίζουμε μια σειρά από συναρτήσεις για την εκτέλεση επί μέρους λειτουργιών. Συγκεκριμένα:

- **coordinates( )**: Λαμβάνει μια καταχώρηση του dataset (διαδρομή) και επιστρέφει τις συντεταγμένες επιβίβασης.
- **haversine( )**: Λαμβάνει 2 ζεύγη συντεταγμένων και επιστρέφει τη μεταξύ τους απόσταση Haversine [2].
- **closest( )**: Λαμβάνει τα κέντρα και ένα ζεύγος συντεταγμένων, τις οποίες αντιστοιχεί στο κοντινότερό τους κέντρο.
- **sum_by_elem( )**: Λαμβάνει δύο ζεύγη (συντεταγμένες, αριθμός) και αθροίζει κατά μέλη (reduce function).
- **avg_by_elem( )**: Λαμβάνει ένα ζεύγος (συντεταγμένες, αριθμός) και διαιρεί με τον αριθμό (reduce function).

Στο κύριο μέρος του προγράμματος, αρχικά διαβάζουμε τα δεδομένα από το HDFS σε 50 partitions και μέσω mapping της συνάρτησης coordinates εξάγουμε τις συντεταγμένες επιβίβασης (θέσεις 3, 4 για κάθε sample). Έπειτα αρχικοποιούμε τον αλγόριθμο k-means θεωρώντας κέντρα τα 5 πρώτα ζεύγη συντεταγμένων με ids από 1 έως και 5. Για κάθε μία από τις 3 επαναλήψεις του αλγορίθμου κάνουμε τα εξής: Πρώτον, τροποποιούμε την συνάρτηση closest μέσω της μεθόδου partial, ούτως ώστε να λειτουργεί αποκλειστικά με τα κέντρα που έχουμε ορίσει. Εκτελούμε έτσι μια εργασία map κατά την οποία οι συντεταγμένες του συνόλου μας αντιστοιχίζονται στο κοντινότερό τους κέντρο μέσω αυτής της συνάρτησης.

Για την εύρεση των νέων κέντρων, αρχικά κάνουμε map προσθέτοντας έναν άσσο σε κάθε tuple (κέντρο, συντεταγμένες) έτσι ώστε μέσω της sum_by_elem να αθροιστούν όλες οι συντεταγμένες ανά κέντρο και να υπολογιστεί ο αριθμός τους. Ο τελικός μέσος όρος των συντεταγμένων ανά κέντρο υπολογίζεται από την avg_by_elem, το αποτέλεσμα της οποίας ανατίθεται στο εκάστοτε κέντρο, ανανεώνοντας τις συντεταγμένες του. Μετά και την 3η επανάληψη, τα κέντρα που έχουν τελικώς προκύψει γράφονται με κατάλληλες εντολές στο HDFS σε csv μορφή, ολοκληρώνοντας τη διαδικασία.

3 Ψευδοκώδικας

Θα επιχειρήσουμε τώρα να δώσουμε σχηματικά τον ψευδοκώδικα των εργασιών map και reduce του παραπάνω προγράμματος, ούτως ώστε να ξεκαθαριστεί η λειτουργία τους και ο τρόπος αξιοποίησης του εν λόγω προγραμματιστικού μοντέλου. Το πρόγραμμα μπορεί να διαιρεθεί σε 3 MapReduce εργασίες: Η πρώτη αφορά στην εξαγωγή των συντεταγμένων των σημείων επιβίβασης από το dataset και συνεπακόλουθα στην αρχικοποίηση των centroids. Εφόσον κάθε key εδώ είναι διαφορετικό, η συνάρτηση REDUCE_1 δε θα αποφέρει κάποια μεταβολή και παραλείπεται. Η δεύτερη εργασία αφορά στο πρώτο βήμα του αλγόριθμου k-means, δηλαδή την αντιστοίχιση κάθε ζεύγους συντεταγμένων στο κοντινότερο centroid και εκτελείται μέσω της MAP_2. Πηγαίνοντας προς το δεύτερο βήμα του αλγορίθμου, η REDUCE_2 αθροίζει τις συντεταγμένες που αντιστοιχούν σε κάθε centroid και υπολογίζει το πλήθος τους. Εδώ, επειδή ακριβώς η reduce function sum_by_elem ορίζεται για 2 εισόδους, χρησιμοποιούμε από την συνάρτηση MAP_2 έναν επιπρόσθετο άσσο για να είμαστε σε θέση να υπολογίσουμε το πλήθος αυτών των συντεταγμένων:

```
function MAP_1(key,value)
    //key: route id
    //value: csv line
    contents = value.split(",")
    lng, lat = contents[3:4]
    emit (key,(lng,lat))
end function
```

```
function MAP_2(key,value)
    //key: route id
    //value: (lng,lat)
    //requires: centroids as (id,(lng,lat))
    for cen in centroids: dists <- haversine(cen[1],value)
    closest = argmin(dists)
    emit (closest[0],(value,1))
end function
```

```
function REDUCE_2(key,values)
    //key: centroid id
    //values: list of ((lng,lat),int)
    lngs = sum(values[lng])
    lats = sum(values[lat])
    len  = sum(values[int])
    emit (key,((lngs,lats),len))
end function
```

Τέλος, το δεύτερο βήμα του αλγορίθμου ολοκληρώνεται με ένα mapping που υπολογίζει για κάθε centroid τις νέες συντεταγμένες διαιρώντας με τον αριθμό που είχαμε αποθηκεύσει. Επειδή και εδώ τα keys των κέντρων είναι μοναδικά, η REDUCE_3 δε θα αποφέρει καμία αλλαγή και παραλείπεται.

```
function MAP_3(key,value)
    //key: centroid id
    //value: ((lngs,lats),len)
    c_lng = value[lngs]/value[len]
    c_lat = value[lats]/value[len]
    emit (key,(c_lng,c_lat))
end function
```

4 Αποτελέσματα

Το script που υλοποιεί τα παραπάνω εκτελείται με την εντολή spark-submit και αποθηκεύει τα αποτελέσματα σε csv file στο HDFS. Η εκτέλεση του προγράμματος με τα 50 partitions έδωσε το εξής output (centroids.csv):

```
id,    centroid

1,     [-74.01649716978265, 40.71133214251166]
2,     [-0.050751572133430525, 0.0248675648977084]
3,     [-73.99125959737344, 40.742001710376115]
4,     [-74.00504499173235, 40.7333051411992]
5,     [-73.95703637501325, 40.760927083103596]
```

5 References

[1] https://data.cityofnewyork.us/Transportation/2015-Yellow-Taxi-Trip-Data/ba8s-jw6u

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