Random Sampling for Group-By Queries in Flink

Σωτηρία-Μαρία Κατάρα, 2015030040
Χριστίνα Μανάρα, 2015030174

16/03/20

---

1 Περίληψη

Το Random Sampling χρησιμοποιείται ευρέως για την προσέγγιση επερωτήσεων σε μεγάλες βάσεις δεδομένων, καθώς τα ωφέλη είναι σημαντικά από την χρήση του, μιας και μειώνεται η χρήση των απαιτούμενων πόρων και του χρόνου απόκρισης, με μικρό μόνο κόστος σε ό,τι σχετίζεται με το κόστος προσέγγισης. Με τη μέθοδο αυτή, απαντώνται group-by queries, τα οποία αρχικά ομαδοποιούν τα δεδομένα με βάση ένα ή περισσότερα χαρακτηριστικά και αμέσως μετά για καθεμιά από αυτές τις ομάδες υπολογίζονται οι συναθροιστικές και στατιστικές τιμές (mean, standard deviation κ.λπ.). Το πρόβλημα που ανακύπτει με τα group-by queries, επάγεται στο γεγονός ότι η δειγματοληπτική μέθοδος δεν μπορεί να εστιάσει στη βελτιστοποιήση της ποιότητας μίας μόνο απάντησης, αλλά πρέπει να βελτιστοποιήσει συγχρόνως ένα σύνολο από απαντήσεις (μία για κάθε group). Για αυτό το λόγο, αναπτύσσεται ο αλγόριθμος CVOPT, ο οποίος είναι στην ουσία ένα query sampling framework, που επιστρέφει πολλαπλές απαντήσεις.

2 Dataset

Το Dataset που χρησιμοποιήθηκε κατά την υλοποίηση του απαιτούμενου αλγόριθμου, είναι το OpenAQ. Το τελευταίο αποτελείται από δεδομένα της σύστασης του αέρα για κάποια δεδομένη χρονική στιγμή και για κάποια συγκεκριμένη πόλη. Δηλαδή, οι υπάρχοντες αισθητήρες σε κάποια πόλη συλλέγουν δεδομένα που αφορούν την περιεκτηκότητα διάφορων ρυπογόνων και μη ουσιών στην ατμόσφαιρα, για κάποιο χρονικό διάστημα. Συγκεκριμένα, το Dataset που χρησιμοποιήθηκε αφορά τα στοιχεία που συλλέχθηκαν από τη Βαυαρία από τις 20/11/19 έως τις 18/2/20. Τα επιμέρους attributes των δεδομένων είναι τα εξής: **location**, **city**, **utc**, **local**, **parameter**, **value**, **unit**, **latitude**, **longitude**, **attribution**. Αυτά όμως που μας απασχολούν, είναι οι διαφορετικές τιμές της στήλης *value* σε αντιστοίχιση με τη στήλη *parameter*, με βάση την οποία

πραγματοποιείται η ομαδοποιήση. Στην προκειμένη περίπτωση προκύπτουν δύο ομάδες, καθώς το key (στήλη parameter) διαθέτει τις τιμές pm25, no2 δηλαδή τιμές που υποδηλώνουν τους ρύπους στην ατμόσφαιρα. Επίσης, η στήλη *value* ομαδοποιείται αντιστοίχως ανάλογα με το key. Οι υπόλοιπες στήλες περιέχουν ίδιες τιμές για όλα τα δεδομένα. Κατά την υλοποίηση του κώδικα, για τον καλύτερο έλεγχο της ορθότητας των αποτελεσμάτων, κρίθηκε αναγκαία η μείωση των διαστάσεων του dataset σε λιγότερα δείγματα. Τα τελικά αποτελέσματα που προέκυψαν αφορούν το αρχικό dataset.

3 Kafka & Flink

Προκειμένου να πραγματοποιηθεί η επεξεργασία ενός μεγάλου όγκου δεδομένων, για την οποία είναι επιθυμητή η εξαγωγή μερικών στατιστικών, με βάση των οποίων θα υλοποιηθεί η δειγματοληπτική μέθοδος, είναι αναγκαία η χρήση τόσο του Kafka όσο και του Flink.

3.1 Kafka

Μέσω της κατανεμημένης streaming πλατφόρμας του Kafka, επιτυγχάνεται η ανάγνωση του αρχείου σε μορφή .csv. Στην ουσία τα δεδομένα διαβάζονται γραμμή-γραμμή και αυτά μέσω της πλατφόρφας μετατρέπονται σε ένα stream το οποίο θα αποτελέσει την είσοδο στο Flink. Πιο συγκεκριμένα, μέσω του τερματικού δίδονται οι εντολές, όπως αυτές απεικονίζονται στις εικόνες 1 και 2 παρακάτω.

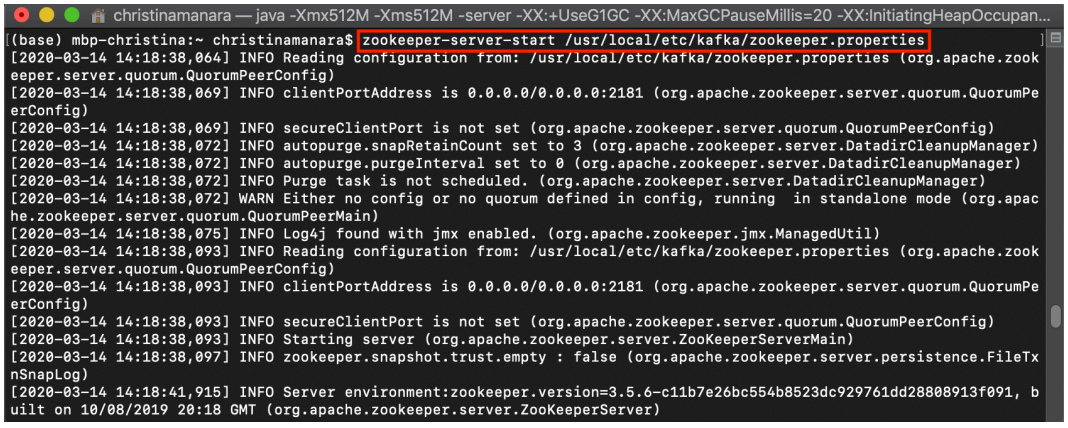

Figure 1: Start Zookeeper

Από την στιγμή που τίθεται σε λειτουργία το Kafka, επόμενο βήμα είναι η δημιουργία των topics. Στην προκειμένη περίπτωση δημιουργούνται αντίστοιχα δύο topics. Το ένα εξ αυτών ονομάζεται *testSource*, το οποίο μεταφέρει τα δεδομένα που θα αποτελέσουν την είσοδο στο Flink. Το άλλο topic ονομάζεται *testSink* και είναι εκείνο, στο οποίο το Flink θα μεταφέρει τα

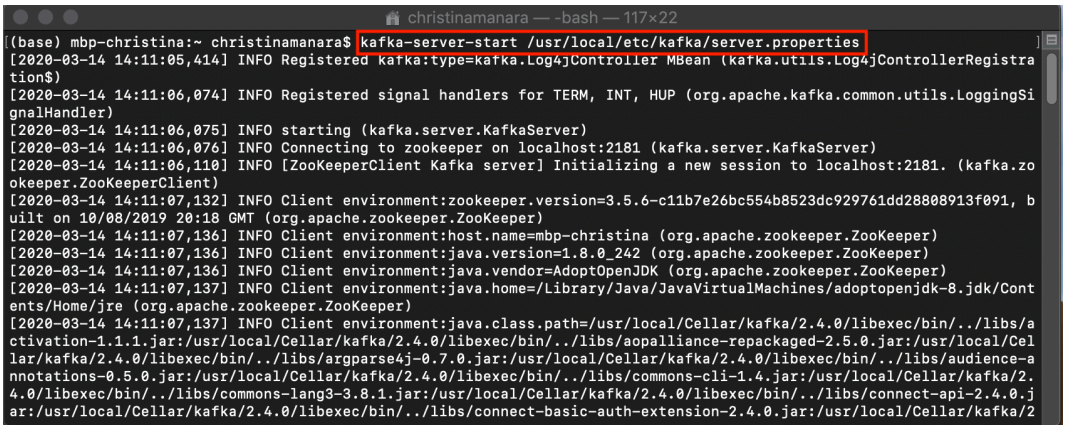

Figure 2: Start Kafka

αποτέλεσματα, που έχουν προκύψει μετά από την απαιτούμενη και κατανεμημένη επεξεργασία των δεδομένων. Εντός των topics, τα δεδομένα χωρίζονται με βάση μία συγκεκριμένη τιμή, αυτή των partitions. Έτσι, για να παραλληλοποιήσουμε την επεξεργασία των δεδομένων η τιμή των partition ορίζεται ίση με δύο (2). Όλα τα παραπάνω υλοποιούνται μέσω της γραμμής εντολών σύμφωνα με την παρακάτω εικόνα.

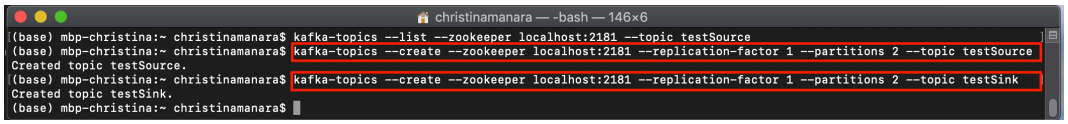

Figure 3: Creation of topics

3.2 Flink

Το Flink, όντας ένα framework, το οποίο επεξεργάζεται κατανεμημένα τα δεδομένα, ώστε να προκύψουν τα επιθυμητά αποτελέσματα γρήγορα και βέλτιστα, αποτελεί το εργαλείο για την υλοποιήση του αλγόριθμου CVOPT, ο οποίος θα αναλυθεί εκτενέστερα σε επόμενη ενότητα. Ειδικότερα, το Flink, δέχεται τα δεδομένα ως ένα stream από το Kafka, και αποτελεί τον καταναλωτή των δεδομένων. Στη συνέχεια, τα αποτελέσματα που προκύπτουν από την εφαρμογή του αλγορίθμου, επιστρέφουν στον Kafka και γράφονται στο *testSink*. Παρακάτω φαίνεται και σχηματικά η δομή της επεξεργασίας των δεδομένων μέσω των Apache Kafka & Flink.

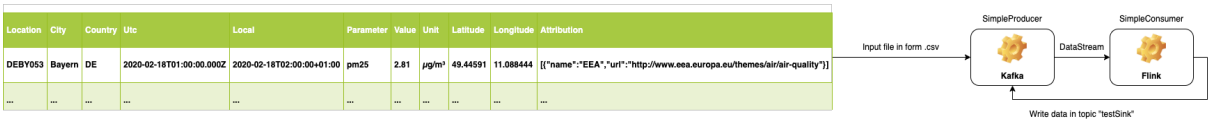

Figure 4: Architecture of system

4 Αλγόριθμος

Ο αλγόριθμος που υλοποιείται σχετίζεται με τον υπολογισμό ενός τυχαίου δείγματος για ένα aggregate και ένα μόνο group-by, δηλαδή όπως αναφέρεται στο paper είναι ο CVOPT-SASG. Η εκτέλεση του περιορίζεται στα ακόλουθα βήματα:

1. Εφαρμογή συνάρτησης flatMap (Tokenizer()) στο input, που διαβάζεται από το testSource του Kafka, με σκοπό την δημιουργία (key,value) pairs σε ένα νέο Datastream (tokenized-Input).

2. Ακολουθεί το πρώτο πέρασμα των δεδομένων. Για κάθε ένα από τα ομαδοποιημένα δεδομένα, που έχουν κατηγοριοποιηθεί (keyBy(0)), υπολογίζονται οι mean και variance τιμές.

   - **Mean Values**: Στα δεδομένα εφαρμόζεται aggregate function για τον υπολογισμό της μέσης τιμής, πάνω σε παράθυρο συγκεκριμένης διάρκειας.
   - **Variance Values**: Ο υπολογισμός των variance values γίνεται σύμφωνα με τον τύπο $V = E[X^2] - E[X]^2$. Σε πρώτο επίπεδο, οι τιμές του datastream tokenizedInput υψώνονται στο τετράγωνο, μέσω μιας flatMap συνάρτησης, και στη συνέχεια υπολογίζεται η μέση τιμή τους, με τον τρόπο που περιγράφηκε στον υπολογισμό των Mean Values. Σε δεύτερο επίπεδο, υψώνονται στο τετράγωνο οι μέσες τιμές μέσω της ίδιας flatMap συνάρτησης που χρησιμοποιήθηκε και προγενέστερα. Τα δύο datastream που προκύπτουν από τους δύο υπολογισμούς, αφού συννενοθούν (union), αποτελούν είσοδο σε μια νέα Reduce Function η οποία αφαιρεί τα πεδία των values και δίνει ως τελικό αποτέλεσμα τα variance values.
   - **Gamma(i) Values**: Όπως φαίνεται και στον αλγόριθμο, οι τιμές αυτές αφορούν στο αποτέλεσμα της διαίρεσης μεταξύ των τιμών της τυπικής απόκλισης και της μέσης τιμής. Οι τιμές τις τυπικής απόκλισης προκύπτουν ως ρίζα των variance values, μέσω μιας flatMap συνάρτησης με εισαγόμενο datastream εκείνο των variance τιμών. Εν συνεχεία οι μέσες τιμές και οι τυπικές αποκλίσεις συννενώνονται σε ένα κοινό datastream. Στο νέο datastream εφαρμόζεται μια Reduce Function, η οποία διαιρώντας τις τυπικές αποκλίσεις με τις μέσες τιμές, εξάγει τις επιθυμητές gamma τιμές. Αξίζει να σημειωθεί ότι η τιμή της μεταβλητής W έχει οριστεί ως ένα (1), διότι στο paper

γίνεται αναφορά ότι τα weights μπορούν να λάβουν αυτή την τιμή λόγω της απουσίας βαρών εισόδου χρήστη (υποσημείωση 3).

- **Stable Gamma Value**: Οι παραπάνω τελευταίες τιμές αθροίζονται μέσω μιας Process Function και προκύπτει η τελική τιμή που αντιστοιχεί στην μεταβλητή γ , η οποία με την σειρά της χρησιμοποιείται κατά την δειγματοληψία.

3. Σε επόμενο επίπεδο ακολουθεί το δεύτερο πέρασμα των δεδομένων. Για καθένα από τα διαφορετικά stratum υπολογίζεται και μια διαφορετική τιμή ως εξής:

   - **S(i) Values**: Προκειμένου να υπολογιστεί η τιμή S(i) για κάθε ομάδα δεδομένων, πραγματοποιείται ένωση των datastream των τιμών Gamma(i) και του Stable Gamma Value. Στο Connected datastream πραγματοποιείται μία Process Function, κατά την οποία εντοπίζεται η τιμή Stable Gamma Value μέσω μίας επαναληπτικής δομής και εν συνεχεία η τιμή αυτή εκχωρείται σε μία τοπική μεταβλητή, η οποία χρησιμοποιείται για την διαίρεση των τιμών Gamma(i) με αυτή την σταθερή τιμή.
   - **Sampling**: Με βάση τον υπολογισμό των τιμών S(i), οι οποίες στρογγυλοποιούνται στο κάτω όριο των τιμών, χρησιποιούνται στη συνάρτηση Sampling Function προκειμένου να προκύψει ο αριθμός των επιλεγμένων δειγμάτων για κάθε ένα stratum. Επίσης, στο σημείο αυτό υλοποιείται και το δεύτερο πέρασμα των δεδομένων, δηλαδή χρησιμοποιείται το αρχικό datastream πάνω στο οποίο καλείται η Sampling Function με όρισμα τις στρογγυλοποιημένες τιμές που έχουν προκύψει προηγουμένως.

Algorithm 1: `CVOPT-SASG`: Algorithm computing a random sample for a single aggregate, single group-by.

Input: Database Table T, group-by attributes A, aggregation attribute d, weight vector w, memory budget M.
Output: Stratified Random Sample S

Let $\mathcal{A}$ denote all possibilities of assignments to A that actually occur in T. i.e. all strata. Let r denote the size of $\mathcal{A}$, and suppose the strata are numbered from 1 till r
For each $i = 1 \dots r$, compute the mean and variance of all elements in stratum i along attribute d, denoted as μ_i, σ_i respectively. Let $\gamma_i \leftarrow \sqrt{w_i}\sigma_i/\mu_i$
$\gamma \leftarrow \sum_{i=1}^{r} \gamma_i$
for $i = 1 \dots r$ **do**
    $s_i \leftarrow M \cdot \gamma_i/\gamma$
    Let S_i be formed by choosing s_i elements from stratum i uniformly without replacement, using reservoir sampling
return $S = [S_1, S_2, \dots, S_r]$

Figure 5: Algorithm of system

5 Αποτελέσματα

Παρακάτω παρουσιάζονται τα αποτελέσματα, όπως αυτά προέκυψαν, από το Eclipse στην εικόνα 6.

```
Mean Values:1> ((no2),21.005826086956514)
Mean Values:1> ((pm25),10.56486956521739)
Numder of Samples is: 4
Numder of Samples is: 28
Sample:2> (no2,8.1)
Sample:1> (no2,11.15)
Sample:2> (pm25,3.32)
Sample:1> (pm25,10.85)
Sample:2> (no2,8.1)
Sample:1> (no2,11.15)
Sample:2> (no2,13.25)
Sample:1> (no2,14.05)
Sample:2> (no2,26.43)
Sample:1> (no2,44.43)
Sample:2> (no2,43.35)
Sample:1> (no2,13.27)
Sample:2> (no2,13.57)
Sample:1> (no2,29.32)
Sample:2> (no2,15.15)
Sample:1> (no2,8.59)
Sample:2> (no2,7.18)
Sample:1> (no2,55.22)
Sample:2> (no2,10.04)
Sample:1> (no2,11.36)
Sample:2> (no2,21.06)
Sample:1> (no2,15.06)
Sample:2> (no2,17.02)
Sample:1> (no2,16.37)
Sample:2> (no2,31.37)
Sample:1> (no2,0.0)
Sample:2> (pm25,11.38)
Sample:1> (pm25,8.59)
Standard Deviation:1> ((pm25),25.54689954353237)
Standard Deviation:2> ((no2),16.359992557806795)
```

Figure 6: Results

6 Συμπεράσματα

Συμπερασματικά, διαπιστώνουμε ότι μετά το πέρας της υλοποιήσης αυτού του αλγορίθμου, πραγματώνεται η εξοικείωση με τόσο με το Apache Kafka όσο και με το Flink. Επίσης, επιτυγχάνεται η επεξεργασία ενός dataset με μεγαλύτερο όγκο δεδομένων, καθώς και η εξαγωγή του αριθμού του δειγμάτων που επιλέχθηκαν με βάση την στρογγυλοποιημένη τιμή των S(i). Προκειμένου να επιτύχουμε καλύτερη απόδοση αυξήσαμε το επίπεδο παραλληλισμού σε δύο (2), παρατηρώντας γρηγορότερη εξαγωγή αποτελεσμάτων. Αυτό πραγματοποιήθηκε με τη δημιουργία δύο (2) partitions σε κάθε topic στον Kafka, καθώς και τη ρύθμιση της μεταβλητής περιβάλλοντος στο Flink, με βαθμό παραλληλοποιήσης ίσο με δύο (2) (StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment().setParallelism(2);) . Τέλος, δοκιμάστηκε η αύξηση του επιπέδου παραλληλισμού στο μέγιστο δυνατό, πειραματισμός που αποδείχθηκε επιζήμιος για την απόδοση του συστήματος, καθώς κάποιες καταστάσεις διατηρούν εσωτερικά δεδομένα ανάλογα του αριθμού των key-groups, γεγονός που οδηγεί στο overloading του συστήματος.

Παραπομπές

1. http://kafka.apache.org/intro

2. https://flink.apache.org/flink-architecture.html