Ειδικά Θέματα Βάσεων Δεδομένων

Skyline Queries Computation - Angle Partitioning

Κωνσταντίνος Μυλωνάς - 2018030151
Παναγιώτης Μίχας - 2015030057

15 February 2023

Οδηγίες

Υλοποιήθηκαν όλα τα ζητούμενα. Στον MR-GRID & MR-DIM αξιοποιούνται όλες οι διαστάσεις για το partitioning. Δεν γίνονται ελεγχοι για ορθά ορίσματα.
Τα topics πρεπει να έχουν δημιουργηθεί:
—bootstrap-server localhost:9092

./bin/flink run ./projects/myartifactid-mysnapshot.jar --a dim --v 1000000000 --p 8 --w 4 --d 2 --it input-topic --tt trigger-topic --ot output-topic

—a: [“dim”, “grid”, “angle”]

—w: parallelism

—v: maximum value (exclusive)

—d: data dimensions

—p: number of partitions: Εξηγείται παρακάτω αναλυτικά

—it: input topic

—ot: output topic

—tt: trigger topic

Για τα πειράματα έχουμε φτιάξει ένα άλλο πρόγραμμα που στην ουσία είναι ένας Kafka-producer. Εκει θέτουμε κάποιες παραμέτρους οπως το topic, το upper bound των τιμών, τον αριθμό των διαστάσεων κλπ. Τα δεδομένα που παράγονται είναι String με αριθμούς που χωρίζονται με κενό. Για παράδειγμα για το data point (2,5,8) αυτό που αντιστοιχεί είναι ενα String “2 5 8”.

Περιγραφή

Συνοπτικά, το πρόγραμμα μας διαβάζει δεδομένα με τη μορφή που είπαμε παραπάνω. Από αυτά τα δεδομένα παράγει στην ουσία τα data points που στη συνέχεια επεξεργάζεται. Πιο συγκεκριμένα, μέσα στο πρόγραμμα επιλέξαμε να αναπαραστήσουμε τα δεδομένα μας ως List<Integer>. Έτσι το "2 5 8" μετατρέπεται σε ένα List<Integer> με πεδία 2,5 και 8. Ο λόγος που κάναμε αυτή την επιλογή είναι επειδή θεωρήσαμε πως οι διαστάσεις των data είναι μεταβλητές. Δηλαδή στο πρόγραμμα μπορούν να δοθούν δεδομένα με οποιονδήποτε αριθμό διαστάσεων και όλες οι διαστάσεις θα ληφθούν υπόψη για τον υπολογισμό του skyline. Εναλλακτικά θα μπορούσαμε να έχουμε πει ότι τα δεδομένα εισόδου θα έχουν σταθερά 10 διαστάσεις για παράδειγμα, και έτσι να έχουμε Tuple10 και κάθε φορά να λαμβάνουμε υπόψη τον αριθμό των διαστάσεων d <= 10 που δόθηκε ως όρισμα. Πήγαμε με την πρώτη προσέγγιση μιας και φαίνεται πως χρησιμοποιούμε λιγότερο χώρο μιας και δεν κρατάμε στη μνήμη τις "περιττές" διαστάσεις.

Αφού μετατραπεί ένα data point στην αναπαράσταση που θέλουμε (List<Integer>) στη συνέχεια υπολογίζουμε το key του με βάση τον εκάστοτε αλγόριθμο και τις λοιπές παραμέτρους. Εξηγούμε τη διαδικασία μεσω ενος παραδείγματος για ευκολία:

Grid (-a)

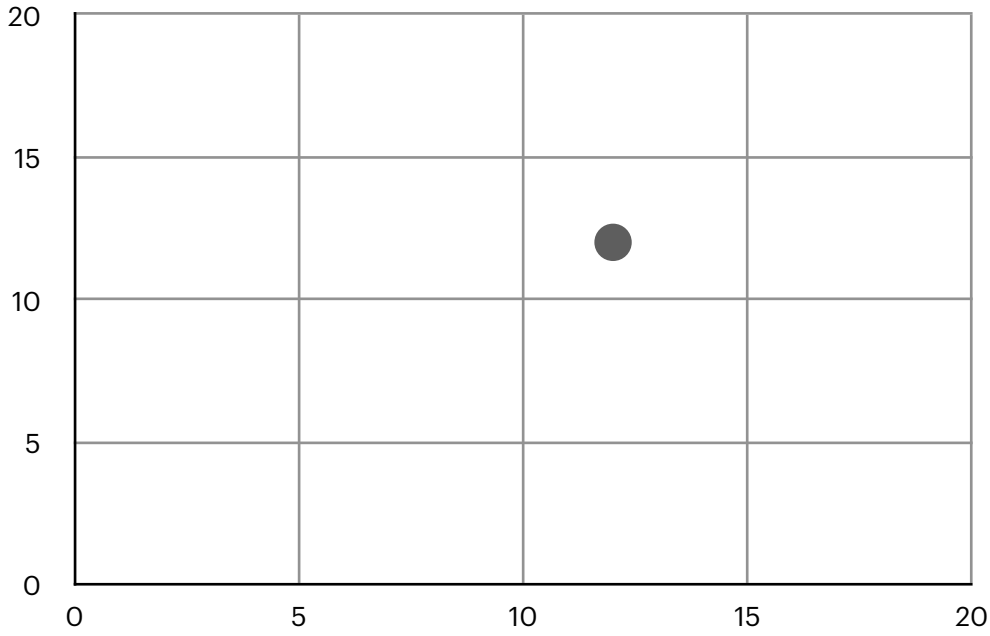
Data Point: (12,12)

Upperbound (-v): 20

Partitions per dim (-p): 4

$$range = \frac{20}{4} = 5$$

$$n_1 = \frac{x_1}{5} = \frac{12}{5} = 2.4 = 2\ integer$$



$$n_2 = \frac{x_2}{5} = \frac{12}{5} = 2.4 = 2\ integer$$

Έχοντας αυτά τα νούμερα, μπορούμε χρησιμοποιώντας το τετραδικό σύστημα αρίθμησης να τα μετατρέψουμε σε decimal και να έχουμε εναν μοναδικό αριθμό για το συγκεκριμένο σημείο. Αυτός ο αριθμός θα είναι ίδιος για οποιοδήποτε άλλο data point πεσει στο συγκεκριμένο cell του grid γιατι τα ακέραια μέρη των n_1, n_2 θα είναι πάντα 2. Έτσι λοιπόν για το συγκεκριμένο σημείο παίρνουμε:

$$n_1 \cdot 4^1 + n_2 \cdot 4^0 = 2 \cdot 4 + 2 \cdot 1 = 10$$

Και πράγματι, αν μετρήσουμε τα partitions ξεκινώντας από το 0, από κάτω προς τα πάνω και από αριστερά προς δεξιά θα δούμε ότι πρόκειται για το 10ο partition. Η διαισθητική εξήγηση είναι ότι το $n_1 = 2$ συμβολίζει πως "έχω προσπεράσει 2 κατακόρυφα partitions δηλαδή δυο 4άδες αφου εχω 4 συνολικά partitions για κάθε κατακόρυφο partition. Το $n_2 = 2$ συμβολίζει ότι είμαι στο 3ο sub-partition του κατακόρυφου partition". Προφανώς θα μπορούσαμε να έχουμε δώσει τα βάρη ανάποδα. Αρκεί να σκεφτούμε ότι

φέρνουμε “τούμπα” το γράφημα. Θα είναι απολύτως ισοδύναμο. Όλα αυτά ισχύουν και για περισσότερες διαστάσεις. Το τετραδικό σύστημα είναι λόγω του ότι χωρίσαμε στα 4 την εκάστοτε διάσταση. Το προσαρμόζουμε ανάλογα με τη παράμετρο. Τα ίδια περίπου ισχύουν και για το angle με πολυ μικρές διαφορές.

Αφού λοιπόν βρούμε το key το κάνουμε append στο τέλος του data point, σαν επιπλέον διάσταση (αλλη μια ευκολία της List). Ετσι στη συνέχεια κάνουμε keyBy με βάση το τελευταίο στοιχείο της λίστας, και μάλιστα είμαστε σίγουροι οτι τα πιθανά κλειδιά είναι:
{0, 1, 2...,t-1}, όπου t είναι ο συνολικός αριθμός των partitions και εξηγούμε παρακάτω πως υπολογίζεται.

Το να ξέρουμε ακριβώς ποια είναι τα κλειδιά είναι πολύ σημαντικό για την υλοποίηση μας όπως θα δούμε παρακάτω.

Το ένα stream με τα data points και το stream για το triggering ενώνονται με connect. Στη συνέχεια ακολουθεί η διαδικασία που περιγράψαμε και γίνεται το partitioning. Τίποτα δεν τυπώνεται μέχρι να έρθει trigger. Ωστοσο εμεις εκμεταλλευόμαστε αυτόν τον χρόνο και υπολογίζουμε τα LocalSkylines του κάθε partition. Το κάθε LocalSkyline είναι αποθηκευμένο σε ένα state variable ListState<List<Integer>> δηλαδή μια λιστα απο λιστες το οποίο by default είναι στο heap εκτος αν ενεργοποιήσουμε state backend RockSB. Ο κάθε worker που μπορεί να χειρίζεται πολλα partitions, ξερει να τα επεξεργάζεται οπως λεγεται συχνά στην ορολογία του flink in keyed-context, δηλαδή ανεξάρτητα το ένα από το άλλο. Όσο φτάνουν νέα δεδομένα ανανεώνεται το local skyline. Η ροή του αλγορίθμου μας είναι η εξης:

Για κάθε νέο data point (s1) το συγκρίνουμε ενα προς ενα με οσα είναι στο LocalSkyline μεχρι στιγμής (s2). Αν s1 > s2 τότε αφαιρούμε το s2 απο το LocalSkyline και συνεχίζουμε το iteration (continue), καθως το

s1 μπορεί να κάνει knock-out και άλλα υποψήφια local-skyline data points. Αν s1 !> s2 τότε πρέπει να δούμε μήπως s2 > s1. Σε αυτή τη περίπτωση “μαρκάρουμε” το s1 ως dominated και βγαίνουμε με break γιατί αν έστω και ένα data point κάνει dominate στο νεο data point, τοτε το νεο data point αποκλείεται να έιναι μέρος του skyline. Τέλος αν s1 !> s2 && s2 !>s1 τότε για λόγους πληρότητας ελέγχουμε αν s1 == s2 γιατί δεν θέλουμε duplicates στο skyline. Αν φτάσει το s1 να συγκριθεί με όλα και δεν είναι dominated, τότε το προσθέτουμε στο state του LocalSkyline, ως υποψήφιο μέρος του. Στη συνέχεια αυτό μπορεί να αλλάξει με την άφιξη νέων data points.

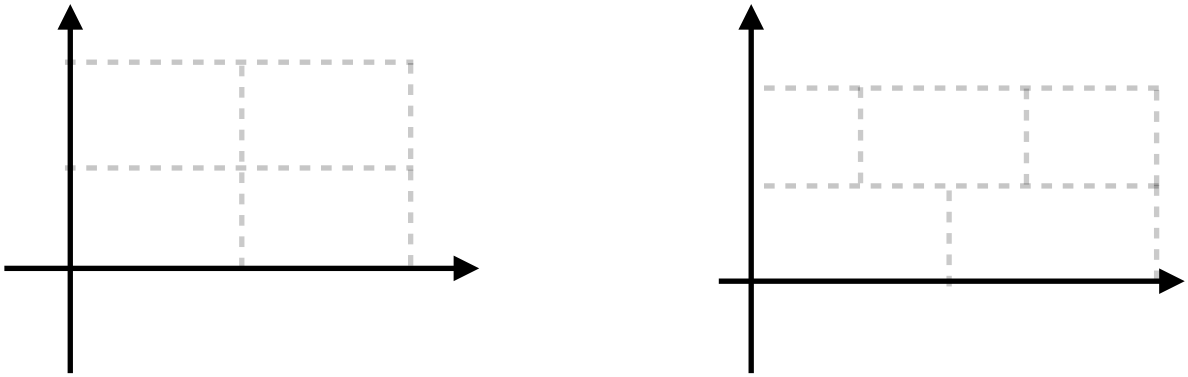
Όλα αυτά όπως είπαμε δεν προωθούνται στο pipeline μεχρι να ερθει trigger. Η δυσκολία με τον trigger είναι οτι ειναι ενας ενω εμείς θέλουμε να ενημερώσουμε όλα τα partitions οτι “ήρθε trigger, ξεκίνα να κάνεις emit το LocalSyline που υπολόγιζες τόση ωρα”. Για τον λόγο αυτόν κάνουμε flatMap και παράγουμε τόσους triggers όσα είναι τα συνολικά partitions (1 trigger / key ή ισοδύναμα 1 trigger / partition). Εδώ φαίνεται η σημασία του να ξέρουμε ποιες τιμές παίρνουν τα keys οπως είπαμε πριν. Σε κάθε trigger δίνουμε το επόμενο στη σειρά κλειδί και ετσι η keyBy το κατατάσσει στο επόμενο partition. Το αποτέλεσμα είναι να γίνονται emit ολα τα LocalSkylines. Βέβαια τελικά βρήκαμε και έναν άλλον τρόπο για να κάνουμε broadcast τον trigger. Πρόκειται για το Broadcast State Pattern αλλα η υλοποίηση είχε προχωρήσει αρκετά για να το αλλάξουμε.

Στη συνέχεια για να παραχθεί το global, πρέπει ολα να καταλήξουν στον ιδιο worker, σε ενα κοινό partition επομένως κάνουμε keyBy με μια fixed τιμή πχ 1. Τα local skylines ρεουν με streaming τροπο στον operator για το global επομένως για να βγάλουμε σωστό global skyline πρεπει να περιμένουμε ωστε ολα τα data points απο ολα τα local skylines έχουν φτάσει και έχουν ληφθεί υποψη. Αυτο επιτυγχάνεται στην ουσία περιμένοντας να φτάσουν όλοι οι triggers. Οι triggers γίνονται emit

τελευταίοι απο το εκαστοτε local skyline αρα όταν φτάσει και ο τελευταίος trigger, σίγουρα έχουν φτάσει ολα τα δεδομένα. Τότε γράφουμε το skyline στο output topic.

Εξήγηση κάποιων λεπτομερειών

Για τον MR-DIM ο αριθμός των συνολικών partitions δίνεται ως όρισμα. Τα πράγματα εδω είναι απλά καθώς χρησιμοποιούμε μόνο τη μια διάσταση για να κάνουμε το partitioning. Ωστοσο στους άλλους δυο αλγορίθμους είναι λίγο πιο πολύπλοκο, δεδομένου ότι χρησιμοποιούνται όλες οι διαστάσεις. Για παράδειγμα στον grid:
Αν πούμε ότι δίνεται σαν όρισμα ο αριθμός των συνολικών partitions τότε ελοχεύει ο εξης κίνδυνος. Έστω 2 διαστάσεις. Αν το όρισμα είναι 5 partitions συνολικά τότε χαλάει η συμμετρία που θα είχαμε με 4 partitions συνολικά, όπως φαίνεται στο παρακάτω σχήμα.



Στην περίπτωση λοιπόν που δεν έχουμε συμμετρία δεν μπορούμε να ισχυριστούμε πως το κάτω αριστερά partition κάνει dominate το πάνω δεξιά. Η σχέση dominance ανάμεσα σε partitions, είναι ο κύριος που τον καθιστά καλύτερο απο τον MR-DIM επομένως πρέπει αν μπορούμε να την

επικαλεστούμε.
Έχοντας αυτά κατα νου, καθώς και για λόγους ευκολίας στην σχεδίαση, αποφασίσαμε το όρισμα “partitions” που δίνεται στην περίπτωση του MR-GRID και του MR-ANGLE να αναπαριστά τον αριθμό των partitions ανα διάσταση, εννοώντας σε πόσες “φέτες” κόβεται η κάθε διάσταση.
Με λίγα μαθηματικά εύκολα καταλήγει κανείς στους παρακάτω τύπους για τον αριθμό των συνολικών partitions:

$$MR-DIM: t = p^d,$$
$$MR-ANGLE: t = p^{d-1}$$

t: συνολικός αριθμός partitions,
p: partitions ανα διάσταση (ορισμα του προγράμματος)
d: αριθμός διαστάσεων

Με αυτόν τον τρόπο είμαστε σίγουροι ότι όλες οι διαστάσεις χωρίζονται ομοιόμορφα, όμως δεν μπορούμε να έχουμε κάθε στιγμή όσα partitions θέλουμε στο σύνολο. Π.χ. για 3 διαστάσεις δεν μπορούμε να έχουμε 15 partitions συνολικά, καθώς δεν υπάρχει ακέραιος αριθμός p με $p^3 = 15.$

Επίσης στον MR-ANGLE δεν μας φάνηκε χρήσιμο κάπου το να υπολογίσουμε το μέτρο καθώς οι γωνίες είναι αρκετές για να κάνουμε το partitioning. Στη συνέχεια χρησιμοποιούνται οι καρτεσιανες συντεταγμένες για τον υπολογισμό του skyline.

Τέλος για τον MR-GRID θεωρούμε πως το καλύτερο partition (αυτο που κάθεται στην αρχή των αξόνων και κάνει dominate οτιδήποτε είναι πανω και δεξιά του) εχει τουλάχιστον ένα data point. Αυτό το αναφέρουμε γιατι στην

περίπτωση που δεν υπάρχει data point σε αυτό το partition τότε δεν πρέπει να κόψουμε ολα τα πανω δεξια partitions ως dominated. Ιδανικά πρέπει να ελέγξουμε ποιο είναι το επόμενο dominant partition που έχει data point κλπ αλλά αυξάνεται αρκετά η πολυπλοκότητα και επίσης χρησιμοποιούμε τόσα πολλά δεδομένα στα πειράματα που ειναι σχεδον απιθανο να μην πέσει κανένα στο partition 0.

Πειράματα

Τα πειράματα έγιναν σε Macbook Air 2020 (M1, 8GB).
Flink 1.16

Οι χρόνοι που αναγράφονται είναι milliseconds και αφορούν τον χρόνο απο τη στιγμή που φτάνει το 1ο data point στον global skyline operator μεχρι τη στιγμή που αυτός ο operator παραλαμβάνει και το τελευταίο trigger, γεγονός που σημαίνει οτι έχει υπολογιστεί το global skyline. Παρακάτω εξηγούμε το πως καταλήξαμε σε αυτή την απόφαση.

Κατα τη διεξαγωγή των πειραμάτων ένας παράγοντας που φάνηκε να πειράζει αισθητά την απόδοση των αλγορίθμων, ήταν ο συνολικός αριθμός των partitions. Σύμφωνα με οσα έχουμε πει δεν μπορούμε να ελεγχουμε πλήρως αυτόν τον αριθμό σε κάθε περίπτωση. Επίσης, δεν γνωρίζουμε ποιος αριθμός partitions είναι ο βέλτιστος για τον κάθε αλγόριθμο, για την κάθε διάσταση κ.ο.κ. Αυτό γίνεται πιο φανερό στον ακόλουθο πίνακα, όπου ως όρισμα -p (partitions) δώσαμε 4, και αριθμό παραλληλισμού επίσης ίσο με 4. Με βάση τους προηγούμενους τύπους, για τις 3 διαστάσεις αυτό μεταφράζεται ως εξής

$$MR - DIM : t = 4$$

$$MR - GRID : t = 64$$

$$MR - ANGLE : t = 16$$

Όσο ανεβαίνουν οι διαστάσεις αυτά τα νούμερα μεγαλώνουν παρά πολύ. Ενδεικτικά για τον MR-GRID στις 5 διαστάσεις έχουμε 1024 partitions.

4 Parallelism - 10.000.000 Data Points - Milliseconds

DIMENSIONS	MR-DIM	MR-GRID	MR-ANGLE
2	52	40	25
3	29	117	84
4	80	293	129
5	320	1871	598

Όπως φαίνεται και από τα αποτελέσματα ο πολύ μεγάλος αριθμός των partitions ρίχνει την απόδοση των αλγορίθμων. Για τον λόγο αυτόν στα υπόλοιπα πειράματα αποφασίσαμε να κρατάμε όσο το δυνατόν πιο κοντά τον συνολικό αριθμό των partitions, μεταξύ των αλγορίθμων. Λέμε όσο το δυνατόν πιο κοντά καθώς όπως έχουμε ήδη αναφέρει, δεν έχουμε πλήρη έλεγχο πάνω σε αυτό.

Στα ακόλουθα πειράματα δίνουμε μεταβλητά ορίσματα -p ανάλογα με τον αλγόριθμο και τη διάσταση με σκοπό να πετύχουμε κοντινούς αριθμούς partitions.

4 Parallelism - 10.000.000 Data Points - Milliseconds - 1

DIMENSIONS	MR-DIM	MR-GRID	MR-ANGLE
2	53	89	67
3	73	49	112
4	106	88	94
5	574	630	279
6	4765	2908	2168

Όπως βλέπουμε εδώ τα αποτελέσματα είναι πιο κοντά σε αυτό που περιμένουμε. Η διαφορά γίνεται πιο αισθητή καθώς ανεβαίνει ο αριθμός των διαστάσεων.

Η αλήθεια είναι πως για την διεξαγωγή των πειραμάτων υπήρξαν κάποια ερωτηματικά, όπως:

1. Ποιός είναι ο χρόνος που τελικά μετράμε; Ο χρόνος που μετρήσαμε είναι ο χρόνος από τη στιγμή που φτάνει το πρώτο data point στο task που υπολογίζει το global skyline, μέχρι τη στιγμή που έχει υπολογιστεί το συνολικό skyline.
2. Πως είμαστε σίγουροι ότι έχει έρθει η κατάλληλη στιγμή για να στείλουμε trigger; Εδω το θέμα μας ήταν πως απο τη στιγμή που τρέχει το πρόγραμμα, συνεχώς καταναλώνονται events απο το kafka topic (μέχρι που κάποια στιγμή τελειώνουν, μιας και στο topic υπαρχει finite αριθμός απο event - στα πειράματα 10.000.000). Όσο έρχονται events υπολογίζονται τα local skylines. Για να μπορούμε να συγκρίνουμε τους αλγορίθμους μεταξύ τους, πρέπει να είμαστε σίγουροι πως τη στιγμή που στέλνουμε το trigger εχει καταναλωθεί ο ίδιος αριθμός από events για τη κάθε περίπτωση. Αυτό που κάναμε τελικά ήταν να περιμένουμε ένα ασφαλές χρονικό διάστημα (1 - 1.5 λεπτο για πολλές διαστάσεις, μερικά δευτερόλεπτα για λίγες) ώστε να είμαστε σίγουροι πως έχει καταναλωθεί όλο το topic και επειτα να στέλνουμε το trigger.

Το ακόλουθο πείραμα έχει να κάνει με την αύξηση του παραλληλισμού. Επιλέξαμε να εξετάσουμε τη συμπεριφορά του MR-Angle καθώς αυξάνεται ο παραλληλισμός (-w flag). Βέβαια εδω δεν περιμέναμε να δούμε φοβερές διαφορές καθώς το τρέχουμε στον δικό μας υπολογιστή, επομένως οι workers έχουν πεπερασμένους πόρους να εκμεταλλευτούν.