Κλιμακωτές χωρικές αποστάσεις για δεδομένα σημείων

Περίληψη

Τα τελευταία χρόνια, η τεχνολογία των συσκευών, αυξάνεται ολοένα και περισσότερο με ραγδαίους ρυθμούς. Οι υπηρεσίες που βασίζονται σε τοποθεσίες αυξάνονται, και αυτό έχει ως αποτέλεσμα τα δεδομένα που περιέχουν γεωγραφικές τοποθεσίες να έχουν την ανάγκη για ευκολότερη αποθήκευση, και αποδοτικότερη επεξεργασία. Η χρήση εφαρμογών που αφορούν χωρικά δεδομένα, με τεράστιες ποσότητες αυτών γίνεται αρκετά δημοφιλής, και καθιστά αναγκαία την επεξεργασία μεγάλων δεδομένων σε κατανεμημένα συστήματα. Πιο συγκεκριμένα, είναι αρκετά διαδεδομένα τα ερωτήματα απόστασης (Distance Join Queries), τα οποία έχουν εφαρμογή στην εξόρυξη δεδομένων, στα πολυμέσα, και στις χωρικές βάσεις δεδομένων. Τέτοιου είδους ερωτήματα, όμως είναι αρκετά δαπανηρά, διότι περιέχουν συνενώσεις (joins) λαμβάνοντας υπόψη τις μετρικές αποστάσεων όπου τίθενται. Τα περισσότερο διαδεδομένα ερωτήματα απόστασης είναι τα *ε*-Distance Join Queries (*ε*DJQ) και τα *K* Closest Pairs Queries (*K*CPQ). Για διευκόλυνση, και αποτελεσματικότερη εκτέλεση τέτοιων ερωτημάτων απόστασης, έγινε απαραίτητη η ανάπτυξη νέων τεχνολογιών για μεγάλα δεδομένα, και κατανεμημένη επεξεργασία αυτών σε clusters υπολογιστών. Τέτοια συστήματα είναι συστήματα όπως το Hadoop και το Spark, όπου επιτρέπουν αυτού του είδους την επεξεργασία χωρικών δεδομένων μεγάλης κλίμακας.

Εισαγωγή

Λόγω της αύξησης των δεδομένων, και της εξέλιξης των εφαρμογών που περιέχουν χωρικές πληροφορίες αναπτύχθηκε η ανάγκη δημιουργίας συστημάτων όπου επιτρέπουν την παράλληλη επεξεργασία δεδομένων. Στη παρούσα μελέτη χρησιμοποιείται το σύστημα Spark για παράλληλες διαδικασίες. Το Apache Spark αποτελεί σύστημα επεξεργασίας δεδομένων που μπορεί να εκτελεί γρήγορα εργασίες σε πολύ μεγάλα σύνολα δεδομένων. Διανέμει εργασίες και τις επεξεργάζεται παράλληλα σε πολλούς υπολογιστές, είτε μόνο του είτε σε συνδυασμό με άλλα κατανεμημένα υπολογιστικά εργαλεία. Στην εργασία αυτή, εφαρμόζεται Distance Join Query (DJQ), το οποίο αξίζει να σημειωθεί ότι έχει μεγάλη χρησιμότητα, καθώς παίζει πολύτιμο ρόλο σε πολλές εφαρμογές της καθημερινής ζωής.

Πιο συγκεκριμένα, δεδομένων δύο μεγάλων συνόλων δεδομένων, A και B ένα Distance Join Query, βρίσκει για κάθε σημείο του συνόλου A, όλα τα σημεία του συνόλου B όπου βρίσκονται στον κυκλικό δίσκο που δημιουργείται δεδομένης ακτίνας ϑ, εφόσον το κέντρο είναι καθένα από τα σημεία του συνόλου A. Συνεπώς, το ερώτημα αυτό βρίσκει όλα τα πιθανά ζεύγη σημείων στο A × B όπου απέχουν απόσταση μικρότερη-ίση του ϑ. Για τους υπολογισμούς των αποστάσεων μεταξύ των σημείων χρησιμοποιείται η Ευκλείδεια απόσταση. Συνεπώς, μπορούν να θεωρηθούν «κοντινά» τα ζεύγη σημείων όπου απέχουν μεταξύ τους Ευκλείδεια απόσταση μικρότερη ή ίση του threshold που έχει τεθεί.

KEYWORDS

Apache Spark, Big Data, Distance Join Query, χωρικά δεδομένα, Ευκλείδεια απόσταση, 2d Grid, αντιγραφή, διαμοιρασμός, παράλληλες διεργασίες, γεννήτρια δεδομένων

Σχετικές Έρευνες

Διάφορες έρευνες έχουν δημοσιευθεί κατά καιρούς όσον αφορά την παράλληλη επεξεργασία για distance join ερωτήματα. Η [1] αφορά την διαχείριση χωρικών δεδομένων κειμένου. Παρουσιάζονται διάφοροι τύποι ερωτημάτων, χαρακτηριστικά ερωτήματα συνένωσης με βάση την απόσταση και αναλύονται τρόποι ποσοτικοποίησης της ομοιότητας. Συγκεκριμένα χρησιμοποιείται η Ευκλείδεια απόσταση για την χωρική ομοιότητα και η Jaccard για την ομοιότητα κειμένου. Χρησιμοποιείται το σύστημα Spark για τον διαμοιρασμό των δεδομένων και την κατάλληλη επεξεργασία τους. Εφαρμόζονται δύο τρόποι για τον διαμοιρασμό των δεδομένων. Αρχικά με οριζόντια κατάτμηση του χώρου, και έπειτα με 2d Grid. Εν κατακλείδι, παρουσιάζονται πειράματα με τους δύο αυτούς τρόπους διαμοιρασμού όσον αφορά τη λειτουργία του συστήματος και τον χρόνο εκτέλεσης.

Η [2] εμβαθύνει σε δύο είδη ερωτημάτων. Το *K* Closest Pair Query (*K*CPQ), όπου βρίσκει τα *K* κοντινότερα ζεύγη σημείων από δύο σύνολα δεδομένων και το *ε*-Distance Join Query (*ε*DJQ), όπου βρίσκει τα ζεύγη σημείων από δύο σύνολα δεδομένων όπου βρίσκονται σε ένα κατώφλι (threshold) απόστασης το ένα από το άλλο. Για τη χρήση των ερωτημάτων αυτών χρησιμοποιούνται τα δύο πλέον κορυφαία κατανεμημένα συστήματα διαχείρισης χωρικών δεδομένων το SpatialHadoop και το LocationSpark, τα οποία και κατά την διάρκεια της εργασίας συγκρίνονται ως προς την απόδοση μεταξύ τους. Γίνονται πειράματα σε πραγματικά σύνολα δεδομένων για να μπορέσει να αξιολογηθεί η απόδοση καθενός αυτά. Εν κατακλείδι, παρατίθεται το συμπέρασμα ότι το Spark είναι πιο αποδοτικό λόγω του μικρότερου χρόνου εκτέλεσης των ερωτημάτων, που οφείλεται στην αποτελεσματικότητα της επεξεργασίας στη μνήμη που παρέχεται από το σύστημα.

Η [3] χρησιμοποιεί ένα Ιδιαίτερα διαδεδομένο σύστημα επεξεργασίας χωρικών δεδομένων, το LocationSpark, το οποίο χτίστηκε πάνω στο Apache Spark. Το σύστημα αυτό προσφέρει ένα χρήσιμο σύνολο λειτουργιών για ερωτήματα απόστασης. Το LocationSpark για να επιτύχει υψηλή απόδοση χρησιμοποιεί διάφορους χωρικούς δείκτες για δεδομένα που βρίσκονται στη μνήμη και εγγυάται ότι οι αμετάβλητοι χωρικοί δείκτες έχουν χαμηλή επιβάρυνση με ανοχή σφαλμάτων.

ε-Distance Join Query

Ένα τέτοιο ερώτημα θέτει το σημείο του ερωτήματος ως το κεντρικό σημείο και ανακτά τα σημεία του δεύτερου συνόλου που βρίσκονται σε απόσταση μικρότερη από *ε*. [4] Από μαθηματική σκοπιά:

Έστω $\mathrm{A} = \{a_0, a_1, \cdots, a_{n-1}\}$, $\mathrm{B} = \{b_0, b_1, \cdots, b_{n-1}\}$ δύο σύνολα από σημεία, και ένα κατώφλι απόστασης $\varepsilon \in \mathbb{R}_{\geq 0}$. Τότε το αποτέλεσμα από το εDJQ είναι ένα σύνολο $\varepsilon DJQ(A, \mathrm{B}, \varepsilon) \subseteq \mathrm{A} \times \mathrm{B}$, περιέχει όλα τα πιθανά διαφορετικά ζεύγη από σημεία στο $\mathrm{A} \times \mathrm{B}$ που έχουν απόσταση μεταξύ τους μικρότερη ή ίση του ε:

$$\varepsilon DJQ(\mathrm{A}, \mathrm{B}, \varepsilon) = \{(a_i, b_j) \in A \times B : \mathrm{dist}(a_i, b_j) \leq \varepsilon\}$$

Δεδομένα

Για την εφαρμογή του ερωτήματος για κοντινά σημεία από δύο σύνολα δεδομένων, αποφασίστηκε η δημιουργία γεννήτριας τυχαίων δεδομένων σε συγκεκριμένα γεωγραφικά πλαίσια. Συγκεκριμένα όσον αφορά το longitude επιλέχθηκαν τιμές στο διάστημα [-30,30) και όσον αφορά το latitude, τιμές στο διάστημα [-30,68). Η υλοποίηση της γεννήτριας τυχαίων τιμών επιτεύχθηκε μέσω της μεθόδου *numpy.random.uniform*. Επιλέχθηκε η παραγωγή 10.000 σημείων για το πρώτο σύνολο δεδομένων και 10.000 σημείων για το δεύτερο σύνολο δεδομένων.

Με τον τρόπο αυτό επιτεύχθηκε η ομοιόμορφη κατανομή στο χώρο "ορθογωνίου παραλληλογράμμου" για καθένα από τα δύο σύνολα δεδομένων. Στη συνέχεια έγινε εξαγωγή των δύο αρχείων που παράχθηκαν σε μορφή *.csv* για την μετέπειτα επεξεργασία, και μετατράπηκαν σε DataFrames (*df1* και *df2*).

Υποδομή

Στη παρούσα εργασία χρησιμοποιήθηκε το cloud *Okeanos* ως υποδομή για το κατανεμημένο σύστημα Apache Spark. Ο cluster στον οποίο διαμοιράστηκαν τα δεδομένα αποτελείται από 4 μηχανήματα (κόμβους). Το ένα από τα μηχανήματα είναι ο master ο οποίος παίζει και τον ρόλο του worker και τα 3 επιπλέον μηχανήματα λειτουργούν ως workers. Συνεπώς τα δεδομένα έχουν συνολικά διαμοιραστεί σε 4 μηχανήματα. Ο master κόμβος έχει 10GB αποθηκευτικό χώρο, 4CPUs και 8GB RAM, καθώς οι υπόλοιποι 3 κόμβοι έχουν από 5GB αποθηκευτικό χώρο, 4CPUs και 8GB RAM ο καθένας.

Περιγραφή Μεθόδου

Τεχνικές Διαμοιρασμού (Partitioning)

Αρχικά, έγινε εισαγωγή των δύο συνόλων δεδομένων, σε DataFrame μορφή με στήλες ID, Latitude, Longitude. Αξίζει να σημειωθεί το γεγονός ότι χρησιμοποιήθηκε η εντολή *cache()* στα δύο DataFrames (*df1* και *df2*).

Η μέθοδος *cache()* είναι μηχανισμός που παρέχεται από το Spark για βελτιστοποίηση με σκοπό την αποθήκευση του ενδιάμεσου υπολογισμού των Spark DataFrames ώστε να μπορούν να επαναχρησιμοποιηθούν σε επόμενες ενέργειες. Πλεονέκτημα της μεθόδου *cache()*, αποτελεί η εξοικονόμηση κόστους για τον λόγο του ότι οι υπολογισμοί είναι πολύ ακριβοί. Επιπλέον, επιτυγχάνεται εξοικονόμηση χρόνου, λόγω της επαναχρησιμοποίησης των επαναλαμβανόμενων υπολογισμών. Τέλος, πλεονέκτημα της μεθόδου *cache()* αποτελεί η εξοικονόμηση του χρόνου εκτέλεσης της διεργασίας.[5]

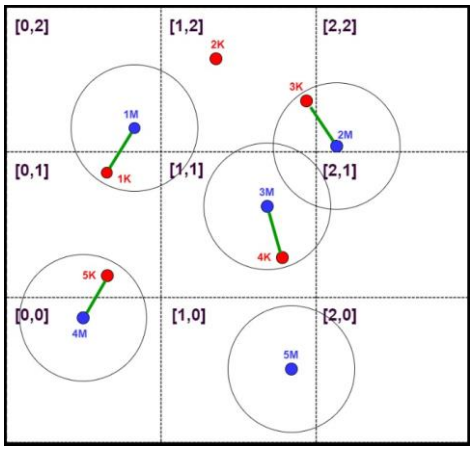
Σε επόμενο στάδιο βρέθηκαν τα:

- min(min_latitude_df1,min_latitude_df2)
- max(max_longitude_df1,max_longitude_df2)

Αυτές οι τιμές χρησιμοποιήθηκαν, ως όρια στο 2d regular Grid. Για τη δημιουργία του 2d Grid (καρτεσιανού πλέγματος δύο διαστάσεων), αποφασίστηκε ο διαχωρισμός του Latitude και Longitude σε 40 ίσα τμήματα (40 Splits). Έτσι κατασκευάστηκαν 40 × 40=1600 παραλληλόγραμμα, τα οποία καλούνται κελιά (cells). Συνεπώς, εκχωρήθηκε η πληροφορία σε ποιο cell ανήκει καθένα από τα σημεία στο δισδιάστατο χώρο.

Στη συνέχεια, ρωτήθηκε από τον χρήστη να δώσει τιμή ϑ, για το κατώφλι απόστασης κοντινών σημείων. Ως αρχική τιμή δόθηκε κατώφλι ϑ=0.4. Έπειτα από την διαίρεση των δύο διαστάσεων (longitude, latitude) με τον αριθμό splits, δημιουργήθηκαν τα *step_lat*, *step_lon* όπου φανερώνουν το μέγεθος των δύο διαστάσεων καθενός από τα κελιά (cells). Συγκεκριμένα, καθένα από τα κελιά έχει διαστάσεις περίπου ίσες με *2.44,1.49* για latitude και longitude αντίστοιχα.

Σε επόμενο στάδιο, αποφασίστηκε να ονομαστούν τα cell ids με μοναδικό τρόπο καθένα ξεχωριστά, όπως φαίνεται και στην παρακάτω εικόνα [*Εικόνα 1*]. Είναι φανερό ότι ζεύγη σημείων τα οποία βρίσκονται στο ίδιο cell έχουν πιθανότητα να είναι κοντινά κατά απόσταση ϑ μεταξύ τους. Όπως φαίνεται στην [*Εικόνα 1]* τα σημεία *3M* και *4K* είναι κοντινά μεταξύ τους και βρίσκονται στο ίδιο cell. Παρόλα αυτά, υπάρχει η πιθανότητα κάποιο ζεύγος να είναι κοντινό κατά απόσταση ϑ, αλλά τα δύο αυτά σημεία να μην ανήκουν στο ίδιο κελί. Τα ζεύγη σημείων *1M* και *1K*, *4M* και *5K*, *2M* και *3K* είναι κοντινά διότι απέχουν απόσταση μικρότερη από ϑ μεταξύ τους, ενώ ανήκουν σε διαφορετικά κελιά.



Εικόνα 1. Παράδειγμα κοντινών σημείων

Λόγω του παραπάνω γεγονότος, τα σημεία του ενός από τα δύο σύνολα δεδομένων αντιγράφηκαν στα γειτονικά κελιά στα οποία μπορούν να θεωρηθούν κοντινά με βάση την απόσταση ϑ. Έτσι, εξασφαλίστηκε η πιθανότητα να έχει καθένα από τα σημεία

κοντινό σημείο σε διαφορετικό κελί. Πιο συγκεκριμένα, αποφασίστηκε η αντιγραφή των σημείων στα κελία όπου επικαλύπτονται από τον κύκλο με κέντρο καθένα από τα σημεία του ενός συνόλου δεδομένων και ακτίνα ίση με ϑ.

Σύμφωνα με την Εικόνα 1, το σημείο *1M*, ενώ ανήκει στο κελί [0,2], σε αυτή τη περίπτωση θα αντιγραφεί στα κελιά [1,2], [1,1] και [0,1], διότι ο κύκλος με κέντρο το σημείο αυτό και ακτίνα ϑ τα επικαλύπτει. Με αντίστοιχο τρόπο, οι αντιγραφές γίνονται και για τα υπόλοιπα σημεία του ίδιου συνόλου δεδομένων.

Συνεπώς, στα πλαίσια της παρούσας εργασίας, βρέθηκαν τα γειτονικά κελιά καθενός κελιού ξεχωριστά. Σε επόμενο στάδιο, έχοντας την πληροφορία των γειτόνων, αλλά και των συνόρων κάθε κελιού με κατάλληλους ελέγχους συνθηκών επιτεύχθηκε η αντιγραφή (duplication) των σημείων στα κελιά. Σημειώνεται, ότι για τα «πάνω», «κάτω», «δεξιά» και «αριστερά» κελία δεδομένου ενός κελιού, η συνθήκη είναι απλούστερη, καθώς ο έλεγχος έχει να κάνει με ένα από τα τέσσερα σύνορα κάθε κελιού. Αντίθετα, για τα «διαγώνια» κελιά, η συνθήκη γίνεται πιο σύνθετη, καθώς συγκρίνεται η απόσταση του σημείου από το γωνιακό σημείο, με την ακτίνα του κύκλου.

Έπειτα από τη διαδικασία της αντιγραφής (duplication) παρατηρήθηκε ότι ο αριθμός των σημείων του αναφερόμενου συνόλου δεδομένων αυξήθηκε σε σημαντικό βαθμό. Ωστόσο, η αντιγραφή αποτελεί ιδιαίτερα χρήσιμη λειτουργία, καθώς εξασφαλίζει την ορθότητα της παράλληλης διεργασίας.

Στη συνέχεια, διατηρώντας για τα δύο DataFrames τις στήλες ID, Latitude, Longitude, cell_id, Dataset, έγινε η ένωσή τους σε ένα DataFrame. Έτσι, έχοντας όλα τα σημεία μαζί, επιτεύχθηκε η διαδικασία του διαμερισμού των δεδομένων στους workers του συστήματος. Επιλέχθηκαν 16 partitions για τον διαμερισμό των δεδομένων, με την εντολή *repartition*. Με επιπλέον δήλωση παραμέτρου τη στήλη cell_id, εξασφαλίζεται η διανομή των σημείων όπου έχουν ίδιο cell_id σε ίδιο partition. Κάθε κελί μαζί με τα σημεία του αποθηκεύεται μοναδικά σε κάποιο από τα partition. Στην [*Εικόνα 2*] απεικονίζεται το πλήθος σημείων όπου έχει καθένα από τα 16 Partition.

partitionId	count
11	2103
2	2243
4	2284
6	2298
14	2319
8	2338
1	2346
7	2411
9	2438
13	2471
10	2533
3	2594
15	2626
0	2691
12	2791
5	2869

Εικόνα 2. Αριθμός σημείων για κάθε PartitionId

Ερώτημα με βάση την απόσταση (εDJQ)

Έπειτα από τη διαδικασία του διαμοιρασμού των δεδομένων στους workers του cluster, πραγματοποιείται το ερώτημα για τα κοντινά ζεύγη σημείων.

Έχοντας το DataFrame με τα χαρακτηριστικά των σημείων των δύο συνόλων δεδομένων, πραγματοποιήθηκε συνένωση (join) των σημείων που ανήκων στο πρώτο σύνολο δεδομένων (*df1*) με τα σημεία του δεύτερου συνόλου δεδομένων (*df2*). Συνεπώς, έγινε καρτεσιανό γινόμενο μεταξύ όλων των σημείων του πρώτου συνόλου δεδομένων με όλα τα σημεία του δεύτερου συνόλου δεδομένων, όπου ανήκουν στο ίδιο cell_id.

Σε επόμενο στάδιο, υπολογίστηκε η Ευκλείδεια απόσταση[6] όλων των συνδυασμών που προκύπτουν (σημείο του *df1* με σημείο του *df2* με ίδιο cell_id) και εκχωρήθηκε σε νέα στήλη distance.

Τελικά, εφαρμόστηκε το φίλτρο της απόστασης, έτσι ώστε τελικά να διατηρηθούν τα ζεύγη σημείων όπου απέχουν απόσταση μεταξύ τους μικρότερη ίση από ϑ. Ως αποτέλεσμα του ερωτήματος, προκύπτουν ζεύγη σημείων με την απόσταση την οποία απέχουν μεταξύ τους. Στη παρακάτω εικόνα [*Εικόνα 2*] φαίνονται ενδεικτικά τα πρώτα 5 ζεύγη σημείων όπου απέχουν μεταξύ τους απόσταση μικρότερη του threshold όπου τέθηκε. Τα ζεύγη αυτά είναι σε πλήθος ίσα με 7675.

df1_id	df2_id	distance
4359	2	0.3963761750626642
3431	2	0.20815422542577738
668	30	0.21688347259624888
4363	75	0.19460564787415166
5054	126	0.22193157838854333

Εικόνα 3. Κοντινά ζεύγη σημείων με απόσταση μικρότερη του threshold ϑ

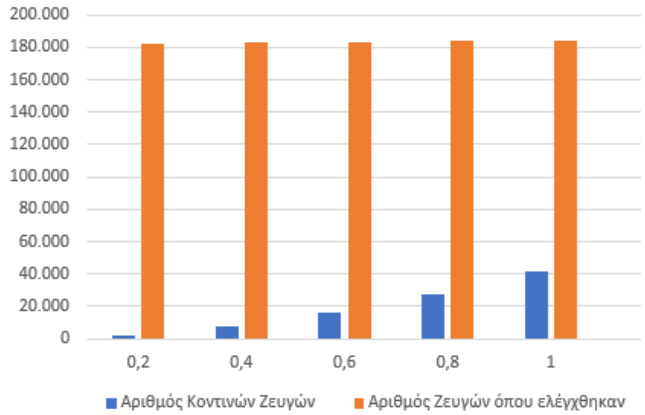
Πειραματική Αξιολόγηση

Πειραματική Αξιολόγηση στην εκτέλεση του Query

Σε αυτό το Κεφάλαιο παρουσιάζονται κάποια πειράματα που υλοποιήθηκαν με αλλαγές σε κάποιες βασικές μεταβλητές του αλγορίθμου. Στα Πειράματα 1 και 2 εφαρμόστηκαν 40 Splits, 16 Partitions και λήφθηκαν διάφορες τιμές για την απόσταση ϑ. Σκοπός, να διαπιστωθεί πως μεταβάλλονται τα κοντινά ζεύγη κατά την εκτέλεση του ερωτήματος καθώς και τα ζεύγη όπου ελέγχθηκαν μετά το join για τον υπολογισμό της απόστασης.

ϑ	Κοντινά Ζεύγη	Ζεύγη όπου ελέγχθηκαν
0.2	1.994	182.691
0.4	7.675	183.141
0.6	16.440	183.522
0.8	27.804	183.641
1	41.152	183.833

Πίνακας 1. Αποτελέσματα ερωτημάτων με αλλαγή στη τιμή ϑ



Εικόνα 4. Ραβδόγραμμα αποτελεσμάτων με αλλαγή της τιμής ϑ

Πείραμα 1

Αρχικά, πραγματοποιώντας δοκιμές στις τιμές της απόστασης ϑ παρατηρείται μεταβολή στον αριθμό των κοντινών σημείων-ζευγών που έχει ως αποτέλεσμα το ερώτημα απόστασης. Στον παρακάτω πίνακα [*Πίνακας 1*] καθώς και στην εικόνα του ραβδογράμματος [*Εικόνα 4*] απεικονίζονται τα αποτελέσματα. Μπορεί να παρατηρηθεί ότι όσο αυξάνεται η τιμή ϑ, αυξάνεται το κατώφλι της συνθήκης κοντινότητας μεταξύ των σημείων. Συνεπώς, έπειτα από το φιλτράρισμα της απόστασης ϑ, τα ζεύγη σημείων που ικανοποιούν το κατώφλι κοντινότητας είναι περισσότερα. Συγκεκριμένα, για ϑ=0.2 τα κοντινά ζεύγη σημείων είναι 1.994, ενώ για ϑ=1 τα ζεύγη σημείων είναι σε πλήθος ίσα με 41.152.

Πείραμα 2

Επόμενο πείραμα αποτελεί ο έλεγχος του αριθμού των σημείων που συνενώθηκαν έπειτα από τη μέθοδο join στα δύο DataFrames. Στον παραπάνω πίνακα [*Πίνακας 1*] και στην εικόνα [*Εικόνα 4*] φαίνονται οι τιμές της απόστασης και τα αποτελέσματα των δοκιμών που εφαρμόστηκαν. Όπως παρατηρήθηκε, με την αύξηση της απόστασης ϑ διαπιστώθηκε αύξηση των ζευγών τα οποία συνενώθηκαν μεταξύ τους, και για τα οποία στη συνέχεια υπολογίσθηκε η απόσταση. Αυτό το γεγονός οφείλεται στην διαδικασία της αντιγραφής των σημείων σε άλλα κελιά. Συγκεκριμένα, για ϑ=0.2 συνενώθηκαν για τον μετέπειτα υπολογισμό της απόστασης 182.691 ζεύγη, ενώ για ϑ=1 συνενώθηκαν 183.833, και συνεπώς περισσότερα, αλλά όχι κατά πολύ.

Πειραματική Αξιολόγηση στο χρόνο εκτέλεσης

Ως επόμενο πείραμα, δοκιμάστηκαν αλλαγές παραμέτρων στο cluster του *Okeanos*, έτσι ώστε να εξαχθούν κάποια συμπεράσματα όσον αφορά τον χρόνο που απαιτεί ο διαμοιρασμός των δεδομένων στα μηχανήματα, όσο και η εκτέλεση του ερωτήματος εύρεσης κοντινών ζευγών.

Πραγματοποιήθηκαν αλλαγές στον αριθμό των Partitions και των Splits για να διαπιστωθεί η επίδραση στο χρόνο εκτέλεσης. Στους παρακάτω πίνακες [*Πίνακας 2*], [Πίνακας 3] φαίνονται τα αποτελέσματα του χρόνου εκτέλεσης για τις συγκεκριμένες μεταβολές.

Partitions	Execution Time (ms)
16	65.449
32	66.821
64	67.887

Πίνακας 2. Χρόνος εκτέλεσης με αλλαγή στον αριθμό των Partitions

Splits	Execution Time (ms)
40	69.545
100	63.209
200	61.566

Πίνακας 3. Χρόνος εκτέλεσης με αλλαγή στον αριθμό των Splits

Αυξάνοντας τις τιμές των Partitions, ο χρόνος εκτέλεσης (λαμβάνοντας τον μέσο όρο όλων τον δοκιμών) αυξάνεται αν και όχι σε σημαντικό βαθμό. Συνεπώς, διαπιστώθηκε ότι περισσότερα partitions επιφέρουν μεγαλύτερη επιβάρυνση στο σύστημα. Ενώ αντίστοιχα, αυξάνοντας τις τιμές των Splits, παρατηρείται ότι ο χρόνος εκτέλεσης μειώνεται. Έτσι λοιπόν μπορεί να εξαχθεί το συμπέρασμα ότι περισσότερα Splits επιφέρουν περισσότερο ομοιόμορφο καταμερισμό.

Συνολικά, δεν υπάρχει ιδιαίτερα σημαντική αλλαγή στον χρόνο εκτέλεσης του αλγορίθμου. Το γεγονός αυτό οφείλεται στην ιδιαιτερότητα του συστήματος Spark, να διαχειρίζεται μεγάλα δεδομένα με τη χρήση παράλληλης διεργασίας στους workers και στα partitions του συστήματος.

Συμπεράσματα

Λαμβάνοντας υπόψη την παρούσα μελέτη, παρατηρήθηκε ότι η αύξηση της απόστασης ϑ στον αλγόριθμο, έχει ως συνέπεια την αύξηση των ζευγών για τα οποία ελέγχεται η απόσταση μεταξύ τους. Μία επιπλέον παρατήρηση αποτελεί το γεγονός ότι η αύξηση της απόστασης ϑ, συνεπάγεται την αύξηση των κοντινών ζευγών, όπου έχει ως αποτέλεσμα το ερώτημα.

Όσον αφορά τον χρόνο εκτέλεσης του αλγορίθμου στο cluster του *Okeanos*, παρατηρήθηκε ότι με αύξηση του αριθμού των Partitions, υπάρχει μία ελάχιστη αλλά όχι σημαντική αύξηση του χρόνου. Ενώ με την αύξηση των Splits παρατηρήθηκε μείωση του χρόνου εκτέλεσης του αλγορίθμου. Συνεπώς, περισσότερα Partitions επιφέρουν μεγαλύτερη επιβάρυνση στο σύστημα. Ενώ περισσότερα Splits επιφέρουν περισσότερο ομοιόμορφο καταμερισμό. Συνολικά, το σύστημα Apache Spark, λόγω της παράλληλης διεργασίας διαχειρίζεται και επεξεργάζεται ιδιαίτερα γρήγορα και αποδοτικά μεγάλους όγκους δεδομένων.

Εκτός της δημιουργίας του 2d Grid, θα μπορούσε να χρησιμοποιηθεί κάποιας άλλης μορφής διαχωρισμός του χώρου, όπως για παράδειγμα οριζόντιος διαμερισμός, ή κάποιο είδος δέντρου. Ως προς την κατανομή των δεδομένων στους κόμβους, παρατηρήθηκε ότι με 16 Partitions και 40 Splits, το σύστημα λειτούργησε αποδοτικά, καθώς επιτεύχθηκε ισομερή κατανομή των δεδομένων.

Η παρούσα εργασία, μελέτησε την αποθήκευση δεδομένων και την εκτέλεση ερωτήματος *ε*-distance join για συνολικά 20.000 δεδομένα. Ωστόσο, το ερώτημα αυτό θα μπορούσε να λειτουργήσει πάνω σε πραγματικά χωρικά δεδομένα. Τέτοιο παράδειγμα θα μπορούσε να είναι δεδομένα με συντεταγμένες φαρμακείων – νοσοκομείων. Άλλα παραδείγματα τα οποία έχουν ήδη ερευνηθεί αποτελούν τα ζεύγη LAKES × PARKS, BUILDINGS × PARKS. [7]

REFERENCES

[1] Parallel Processing of Spatio-textual Similarity Join Query, Pavlopoulos Nikolaos, Athens 2019

[2] A Comparison of Distributed Spatial Data Management Systems for Processing Distance Join Queries, Francisco Garcıa-Garcıa, Antonio Corral, Luis Iribarne, George Mavrommatis, and Michael Vassilakopoulos

[3] LocationSpark: A Distributed In-Memory Data Management System for Big Spatial Data, Mingjie Tang, Yongyang Yu, Qutaibah M. Malluhi , Mourad Ouzzani , Walid G. Aref

[4] A Taxonomy for Distance, Based Spatial Join Queries, Lingxiao Li, Monash University, Melbourne, Australia David Taniar, Monash University, Melbourne, Australia, July-September 2017

[5] Spark DataFrame Cache and Persist Explained — SparkByExamples

[6] Euclidean distance – Wikipedia

[7] Efficient Distance Join Query Processing in Distributed Spatial Data Management Systems, Francisco García García, Antonio Corral, Luis Iribarne, Michael Vassilakopoulos, October 2019

[8] Distance Join Queries of Multiple Inputs in Spatial Databases, Antonio Corral, Yannis Manolopoulos, Yannis Theodoridis, Michael Vassilakopoulos, 2003

[9] Apache Spark Architecture | Distributed System Architecture Explained | Edureka