Παράλληλη υλοποίηση Distance Join ερωτημάτων με τη χρήση του Apache Spark

Περίληψη

Η συνεχής τεχνολογική ανάπτυξη καθιστά ολοένα και περισσότερο αναγκαία τη διαχείριση του τεράστιου όγκου δεδομένων που παράγονται καθημερινά. Η διαχείριση καθώς και η εξαγωγή πληροφορίας από τα παραγόμενα δεδομένα είναι εφικτή με την δημιουργία και εφαρμογή κατάλληλων αλγορίθμων σε συνδυασμό με την αξιοποίηση σύγχρονων εργαλείων διαχείρισης τεράστιου όγκου δεδομένων. Η παρούσα εργασία πραγματεύεται την εύρεση ζευγαριών (α,β) πάνω στο επίπεδο, όπου α ε Α και β ε Β, για τα οποία ισχύει: $d(\alpha,\beta) \leq \theta$, με θ μία παράμετρο που δίνεται από το χρήστη. Ο υπολογισμός όμως όλων των πιθανών ζευγαριών απαιτεί μεγάλο υπολογιστικό κόστος, κάνοντας σχεδόν αδύνατη πολλές φορές την εύρεση τους από ένα μόνο μηχάνημα. Λύση στο πρόβλημα αυτό αποτελεί η μηχανή ανάλυσης δεδομένων Apache Spark με την χρήση χωρικού ευρετηρίου. Η γρήγορη επεξεργασία τεράστιου όγκου δεδομένων και η δυνατότητα παράλληλης υλοποίησης σε συνδυασμό με την κατανομή διεργασιών σε παραπάνω από έναν υπολογιστές ελαχιστοποιεί το χρόνο εκτέλεσης. Ο προτεινόμενος αλγόριθμος αξιοποιεί τις παροχές του Apache Spark και του KD-tree αλγορίθμου για την διαμέριση του χώρου. Κατά την πειραματική διαδικασία μελετήθηκε αρχικά τοπικά και στην συνέχεια σε ένα cluster υπολογιστών με την αξιοποίηση πόρων από το cloud Okeanos με τα αποτελέσματά του να παρουσιάζονται αναλυτικά στην παρούσα έρευνα.

CCS Concepts

• Information systems → Information systems applications → Spatial-temporal systems • Computing methodologies → Parallel and Distributed computing methodologies.

Keywords

Apache Spark, Spatial Spark, Distance Join, KD-tree, Scalable Distance Join, Multi-Threading, Hadoop, Yarn, Big Data, Spatial indices

1. ΕΙΣΑΓΩΓΗ

Στην σημερινή εποχή της ραγδαίας εξέλιξης της τεχνολογίας, η έννοια των Big Data είναι στο προσκήνιο, τα χωρικά δεδομένα αυξάνονται εκρηκτικά και τα κεντρικά συστήματα δεν είναι σε θέση να διαχειριστούν αυτόν τον όγκο δεδομένων. Έτσι προκλήθηκε η ανάγκη δημιουργίας νέων συστημάτων που θα μπορούσαν να επεξεργαστούν δεδομένα παράλληλα. Αυτά τα συστήματα ονομάζονται Παράλληλα/ Κατανεμημένα Συστήματα και παρουσιάζουν καλύτερη απόδοση επεξεργασίας. Ωστόσο, η χρήση αυτών των συστημάτων απαιτεί υπολογιστικούς πόρους λόγω της παράλληλης επεξεργασίας, η οποία γίνεται σε διαφορετικά μηχανήματα ή επεξεργαστές που είναι συνδεδεμένοι.

Με κίνητρο λοιπόν την πιεστική ανάγκη για άμεση πρόσβαση στην πληροφορία, οι μηχανές επεξεργασίας μεγάλων δεδομένων και οι διάφορες τεχνικές προσαρμοσμένες σε χωρικά δεδομένα διαδραματίζουν σημαντικό ρόλο. Ιδιαίτερη έμφαση δίνεται στις τεχνικές ευρετηρίασης-διαμέρισης, για την απόκριση σε χωρικά ερωτήματα, ένα βέλος στην φαρέτρα των αναλυτών για την αποτελεσματική και επεκτάσιμη διαχείριση δεδομένων. Για το λόγο αυτό, ακολουθείται ευρέως και η λογική της διαχώρισης του επιπέδου σε υποσύνολα. Αναλυτικότερα πολλές και διαφορετικές εκδοχές αλγορίθμων δέντρων θα μπορούσαν να βοηθήσουν στο διαχωρισμό του χώρου, με την παρούσα έρευνα να παρουσιάζει την χρησιμοποίηση του αλγόριθμου KD-Tree[1].

Στην συγκεκριμένη έρευνα, χρησιμοποιήθηκε το Apache Spark, μια μηχανή ανάλυσης δεδομένων που παρουσιάζεται στην ενότητα 3 και για την υλοποίηση του προτεινόμενου αλγόριθμου, που παρουσιάζεται στην ενότητα 6, χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python. Ο αλγόριθμος μπορεί να διαχειριστεί σύνολα δεδομένων της μορφής (ID,x,y), όπου ως x ορίζεται το longitude και ως y το latitude ενός σημείου, με το ID να αποτελεί το μοναδικό αναγνωριστικό χαρακτηριστικό κάθε σημείου.

2. ΣΧΕΤΙΚΕΣ ΕΡΕΥΝΕΣ

Πολλοί ερευνητές έχουν ανακαλύψει τρόπους για να αναλύσουν χωρικά δεδομένα με σκοπό να βρουν χρήσιμα συμπεράσματα για σημεία ενδιαφέροντος. Η ύπαρξη πολλών σχετικών ερευνών που αφορούν Spatial Join Queries μεγάλων δεδομένων, συνέβαλλαν στην υιοθέτηση ιδεών με αποτέλεσμα την υλοποίηση του προτεινόμενου αλγόριθμου. Άλλωστε, έχουν δημιουργηθεί και μελετηθεί μέχρι σήμερα πολλοί αλγόριθμοι, παρουσιάζοντας τις τεράστιες δυνατότητες καθώς και την αξία του συγκεκριμένου ερευνητικού πεδίου.

Η χρήση των MBR(Minimum Bounding Rectangles) αποτελεί μια μέθοδο ταχύτερου διαβάσματος δεδομένων, διότι κατά την κατασκευή τους, φορτώνονται στην μνήμη του υπολογιστή. Όπως παρουσιάζεται και στην [2], κατά την δημιουργία ενός MBR, επεκτείνονται όλες οι πλευρές ενός ορθογωνίου με ισόποσο τρόπο, παράλληλα με τους άξονες x, y, έτσι ώστε να περιλαμβάνονται όλα τα δεδομένα που χρειάζονται σε κάθε περίπτωση και να μειώνεται ο «νεκρός χώρος». Στην συγκεκριμένη τεχνική βασίστηκε και η παρούσα έρευνα, όπου η δημιουργία και η επέκταση των ορθογωνίων χρησιμοποιήθηκε, έτσι ώστε να αποφευχθεί η επεξεργασία του δεύτερου συνόλου δεδομένων, καθιστώντας παράλληλα δυνατή την πρόσβαση στην απαραίτητη πληροφορία για την υλοποίηση των επιθυμητών ερωτημάτων. Σημαντικό προς αναφορά είναι ότι χωρίς να χαθεί πληροφορία, επιταχύνεται η εκτέλεση του αλγορίθμου.

Για την δημιουργία των MBR, όπως επισημαίνεται στην [3], ιδανική θεωρείται η χρήση ενός αλγορίθμου KD-Tree. Ο αλγόριθμος KD-Tree, χρησιμοποιείται κυρίως συνδυαστικά σε προβλήματα εύρεσης κοντινότερων γειτόνων και απόστασης[4]. Επιπλέον, όπως αναφέρεται και στην [5], τα Kd δέντρα έχουν χρησιμοποιηθεί στην εξισορρόπηση φορτίων από διάφορες εφαρμογές στις υπολογιστικές επιστήμες και την ανάλυση δεδομένων. Στις υπολογιστικές επιστήμες συγκεκριμένα, τα Kd δέντρα χρησιμοποιούνται προκειμένου να βελτιωθεί η επεκτασιμότητα σε μεγάλες προσομοιώσεις.

Η επεκτασιμότητα σε μεγάλες προσομοιώσεις άλλωστε κρίνεται επιτακτική για το πρόβλημα των distance join ερωτημάτων, το οποίο συναντάται όλο και περισσότερο καθημερινά. Λόγω της παραγωγής υπέρογκων δεδομένων από πολλές εφαρμογές, όπως υπογραμμίζεται και στην [6], η αποτελεσματική επεξεργασία εξαιρετικά μεγάλων όγκων χωρικών δεδομένων οδήγησε πολλούς οργανισμούς να χρησιμοποιούν κατανεμημένα συστήματα

επεξεργασίας. Όπως παρουσιάζει και η [7], τα κατανεμημένα συστήματα που βασίζονται στο Apache Spark επιτρέπουν στους χρήστες να εργάζονται σε κατανεμημένα δεδομένα στη μνήμη, χωρίς να ανησυχούν για τον μηχανισμό διανομής δεδομένων και την ανοχή σφαλμάτων. Στο πλαίσιο αυτό, η παρούσα έρευνα παρουσιάζει έναν αλγόριθμο για το πρόβλημα των distance join ερωτημάτων με τη βοήθεια του Spark και την χρήση του Hadoop ως σύστημα αποθήκευσης δεδομένων[8], υλοποιημένο πάνω σε ένα cluster κατανεμημένων υπολογιστών.

3. APACHE SPARK

Το Apache Spark σύμφωνα με την [9] είναι μια μηχανή ανάλυσης δεδομένων σχεδιασμένη να αποδίδει στην υπολογιστική ταχύτητα και επεκτασιμότητα που απαιτούν τα Big Data. Ως αναλυτική μηχανή δύναται να επεξεργάζεται δεδομένα από 10 μέχρι 100 φορές γρηγορότερα από τις υπόλοιπες εναλλακτικές που υπάρχουν και ανταπεξέρχεται σε πολύ μεγάλα σύνολα δεδομένων για παράλληλο και κατανεμημένο προγραμματισμό. Επεκτείνεται διαμοιράζοντας το φόρτο εργασίας σε μηχανήματα μιας τεράστιας συστάδας από υπολογιστές, με την χρήση παραλληλοποίησης[10]. Επιπλέον περιλαμβάνει APIs (application programming interfaces) για τον προγραμματισμό γλωσσών, όπως Python, Scala, Java και R και βιβλιοθήκες για ποικίλες εργασίες όπως SQL(Spark SQL) [11]. Κατάλληλο για εφαρμογή σε streaming δεδομένα, δεδομένα γραφήματος, μηχανικής μάθησης και σε εφαρμογές τεχνητής νοημοσύνης. Συχνά συγκρίνεται με το Apache Hadoop, και πιο συγκεκριμένα με το MapReduce[12], που είναι η εγγενής μονάδα επεξεργασίας δεδομένων του Hadoop. Η κεντρική διαφορά μεταξύ του Spark και του MapReduce είναι ότι το πρώτο επεξεργάζεται και αποθηκεύει τα δεδομένα στην μνήμη RAM για επακόλουθες διαδικασίες, χωρίς όμως να τα διαβάζει ή να τα γράφει στο δίσκο με αποτέλεσμα την γρηγορότερη ταχύτητα επεξεργασίας.

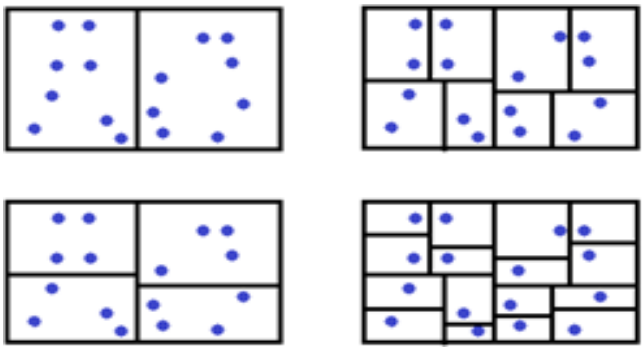
4. ΟΡΙΣΜΟΣ ΠΡΟΒΛΗΜΑΤΟΣ

Το πρόβλημα που απαντάται στην συγκεκριμένη έρευνα είναι η εύρεση ζευγαριών από σημεία στο επίπεδο που βρίσκονται σε μια απόσταση θ και προέρχονται από δύο διαφορετικά σύνολα δεδομένων(distance-join-queries). Την απόσταση έχει την δυνατότητα να την ορίζει ο χρήστης κατά την υλοποίηση του αλγορίθμου. Ως μέτρο απόστασης, για λόγους μείωσης της πολυπλοκότητας των υπολογισμών επιλέχθηκε η ευκλείδεια απόσταση. Αναλυτικότερα όπως αναφέρεται και στο [7] έχουμε ένα σύνολο χωρικών στοιχείων P:{$P0$,$P1$,$P2$,..,PN} και ένα σύνολο χωρικών στοιχείων Q:{$Q0$,$Q1$,$Q2$,..,QM} μαζί με μία απόσταση θ. Το αποτέλεσμα του distance-join-query, θα δώσει όλους εκείνους τους συνδυασμούς σημείων των δύο συνόλων που η απόστασή τους είναι μικρότερη ή ίση με το θ, όπως ορίζεται και στην παρακάτω σχέση.

$$DJ(P, Q, \theta) = \{(pi,qj) \in P \times Q : dist(pi,qj) \leq \theta\} \quad (1)$$

5. ΧΩΡΙΚΗ ΕΥΡΕΤΙΡΙΑΣΗ

Μια ενδιαφέρουσα προσέγγιση για τον χειρισμό της ασυμμετρίας των δεδομένων εισαγωγής, διότι το Apache Spark υστερεί στην υποστήριξη δημιουργίας έξυπνων ευρετηρίων, είναι η χρήση ενός KD δέντρου, του οποίου τα φύλλα αντιστοιχούν σε διαμερίσματα δεδομένων στο κατανεμημένο σύστημα[13]. Αναλυτικά, ως KD – tree ή δέντρο διαστάσεων ορίζεται μια δομή δεδομένων που χρησιμοποιείται για την οργάνωση ορισμένων σημείων σε ένα χώρο με κ διαστάσεις. Τα συγκεκριμένα δέντρα είναι πολύ χρήσιμα για αναζητήσεις εμβέλειας και πλησιέστερων γειτόνων[4]. Αποτέλεσαν μια από τις πρώτες δομές που χρησιμοποιήθηκαν για την ευρετηρίαση σε πολλές διαστάσεις. Κάθε επίπεδο του KD δέντρου χωρίζει τον χώρο σε δύο υπόχωρους, η κατάτμηση γίνεται κατά μήκος μιας διάστασης στο ανώτερο επίπεδο του δέντρου και κατά μήκος μιας άλλης διάστασης στους κόμβους του επόμενου επιπέδου και ούτω καθεξής. Η κατάτμηση προχωρά με τέτοιο τρόπο ώστε σε κάθε κόμβο περίπου το ήμισυ των σημείων που είναι αποθηκευμένα στο υποδένδρο να πέφτουν στη μία πλευρά και το υπόλοιπο ήμισυ των σημείων στην άλλη πλευρά(Εικόνα 1). Με αυτόν τον τρόπο παρέχει έναν γρήγορο τρόπο πρόσβασης σε οποιοδήποτε αντικείμενο εισόδου ανά θέση. Αν και έχουν επινοηθεί πολλές διαφορετικές υλοποιήσεις KD δέντρων, ο σκοπός τους είναι πάντα ιεραρχικά να αποσυντίθεται ο επιλεγμένος χώρος σε σχετικά μικρό αριθμό κυττάρων, ώστε κανένα κελί να μην περιέχει πάρα πολλά αντικείμενα εισόδου.



Εικόνα 1. Διαμέριση χώρου με τη χρήση KD-tree αλγορίθμου

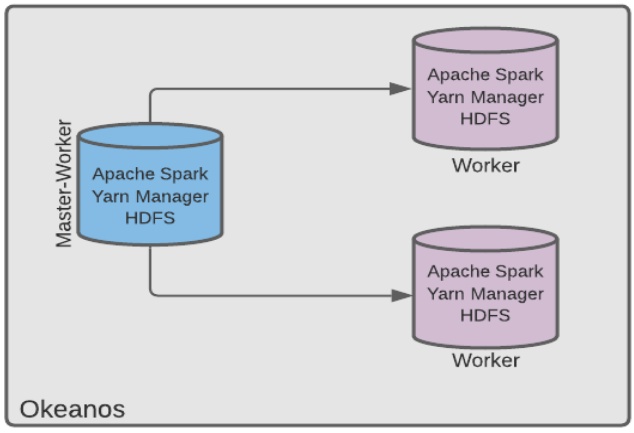
6. ΠΕΡΙΓΡΑΦΗ ΑΛΓΟΡΙΘΜΟΥ

Ο αλγόριθμος κατά την εκκίνηση του διαβάζει τα δύο σύνολα δεδομένων προς ανάλυση και στη συνέχεια εκμεταλλεύεται τα πλεονεκτήματα του αλγορίθμου KD-Tree. Συγκεκριμένα, ο KD-Tree χρησιμοποιεί το ένα εκ των δύο συνόλων δεδομένων χωρίζοντας το επίπεδο σε δύο μέρη, επιλέγοντας μια εκ των δύο διαστάσεων, ξεκινώντας με την διάσταση x. Ορίζει έτσι ώς σημείο διαμέρισης του χώρου, την διάμεσο του ορθογωνίου(box) μέσα στο οποίο βρίσκονται όλα τα σημεία του συνόλου δεδομένων και στη συνέχεια, ο διαχωρισμός γίνεται στη διάσταση του y. Όταν το Kd δέντρο μεγαλώνει σε κόμβους φύλλων m, κάθε κόμβος φύλλων αντιστοιχεί σε μία διαμέριση του χώρου. Η παραπάνω διαδικασία συνεχίζεται μέχρι την επιθυμητή διαμέριση του χώρου. Ο αριθμός των συνολικών διαμερίσεων που πραγματοποιεί ο αλγόριθμος είναι εφικτό να καθοριστεί από τον χρήστη. Στην προκειμένη περίπτωση κρίθηκε ότι η επιλογή 128 διαμερίσεων του χώρου για μεγάλα δεδομένα είναι η αποδοτικότερη και ταχύτερη επιλογή. Η υλοποίηση του αλγορίθμου KD-Tree πραγματοποιήθηκε με την χρήση των τεχνικών Multi-Threading[14]. Με τη δημιουργία των threads, δίνεται η δυνατότητα κατανομής των διεργασιών του προγράμματος ανά thread, παραλληλοποιώντας με αυτό τον τρόπο την διαδικασία διαμέρισης του χώρου. Με την ολοκλήρωση του KD-Tree αλγορίθμου, έχει εξασφαλιστεί επιτυχώς το load balancing λόγω της χρήσης της διαμέσου των δεδομένων. Εν συνεχεία ο αλγόριθμος, χρησιμοποιεί τις συντεταγμένες των διαμερίσεων του χώρου για την δημιουργία ορθογωνίων boxes, τα οποία θα περιέχουν ισομοιρασμένη όλη την πληροφορία του πρώτου συνόλου δεδομένων. Ακολούθως επεκτείνοντας τα παραγόμενα boxes ως προς όλες τις κατευθύνσεις κατά θ, όσο και η απόσταση που έχει ορίσει ο χρήστης για τα distance-join ερωτήματα, εισάγονται με τη βοήθεια της Spark SQL όλα τα σημεία του δεύτερου συνόλου δεδομένων στα εκάστοτε boxes που ανήκουν.

Πριν υπολογιστεί το distance join μεταξύ των δύο συνόλων δεδομένων εκτελείται η σημαντική διαδικασία του repartition[15]. Το Spark με την εισαγωγή των δεδομένων και αναλόγως με το μέγεθος τους, επιλέγει τον αριθμό των partitions με τον οποίο θα χωρίσει τα εκάστοτε δεδομένα. Παρέχεται όμως στον χρήστη η επιλογή του repartition, την οποία εκμεταλλεύεται ο αλγόριθμος διαμερίζοντας εκ νέου τα δεδομένα. Το Spark λαμβάνοντας ως παραμέτρους τις επιθυμητές διαμερίσεις που πραγματοποίησε ο KD-tree αλγόριθμος και τον αριθμό των παραγομένων boxes, χωρίζει τα δύο σύνολα δεδομένων σε 128 partitions[16]. Η συγκεκριμένη διαδικασία δίνει ένα νέο "index" στα σημεία, έτσι ώστε να κατανεμηθούν αναλόγως στα διαθέσιμα μηχανήματα του cluster και να αποφευχθεί η διαδικασία shuffling των δεδομένων. Προκειμένου να γίνει η εύρεση των σημείων που βρίσκονται σε απόσταση θ, πραγματοποιείται το distance join ερώτημα μεταξύ των δύο συνόλων δεδομένων και επιστρέφονται όλα εκείνα τα ζεύγη σημείων που ικανοποιούν τον επιθυμητό έλεγχο. Λαμβάνοντας υπόψιν την υπολογιστική πολυπλοκότητα του συγκεκριμένου ελέγχου και με σκοπό την επιτάχυνση του αλγορίθμου, χρησιμοποιείται ως μέτρο απόστασης το τετράγωνο της απόστασης θ που ορίζει ο χρήστης και τα αποτελέσματα του αλγορίθμου εξάγονται σε αρχεία της μορφής csv για περαιτέρω ανάλυση.

7. CLUSTER

Το Cluster που δημιουργήθηκε στα πλαίσια της εργασίας, αποτελείται από το Apache Spark μαζί με άλλες δύο βοηθητικές μηχανές, το Hadoop Yarn[17] και το HDFS[18]. Ο Yarn αποτελεί τον Resource Manager του Cluster, ο οποίος είναι υπεύθυνος για τη διαχείριση των πόρων του Cluster και την κατανομή των πόρων ανάλογα με τις ανάγκες του εκάστοτε προγράμματος που πρόκειται να επεξεργαστεί το Spark Cluster. Το HDFS αποτελεί το σύστημα διαχείρισης δεδομένων της εφαρμογής, παρέχει ευκολότερη πρόσβαση για τεράστια σύνολα δεδομένων και τα αρχεία αποθηκεύονται σε πολλά μηχανήματα πριν επεξεργαστούν για να σώσουν το σύστημα από πιθανές απώλειες δεδομένων. Το HDFS παρέχει επίσης εφαρμογές για παράλληλη επεξεργασία δεδομένων. Αναλυτικότερα, όπως παρουσιάζεται και στην εικόνα 2, το Spark Cluster έχει 20 CPUs και 18GB Ram διαθέσιμα και αποτελείται από 3 virtual machines, εκ των οποίων ένα έχει τον ρόλο του master-worker και τα υπόλοιπα λειτουργούν ως απλοί workers. Παράλληλα ο master-worker εκτελεί χρέη Namenode – Datanode(HDFS) και Resource Manager(Yarn). Κατά την εκκίνηση λοιπόν κάθε εφαρμογής, ο Spark Driver απευθύνεται στον Yarn Resource Manager για τους απαραίτητους πόρους.



Εικόνα 2. Spark Cluster

8. ΠΕΙΡΑΜΑΤΙΚΗ ΔΙΑΔΙΚΑΣΙΑ

Ο αλγόριθμος που προτείνεται, αρχικά εκτελέστηκε τοπικά, σε stand-alone mode και στην συνέχεια σε ένα Apache Spark Cluster στο cloud Okeanos που υλοποιήθηκε για τις ανάγκες της παρούσας έρευνας. Κατά την πειραματική διαδικασία, ο αλγόριθμος εφαρμόστηκε σε τρία διαφορετικά ζευγάρια συνόλων δεδομένων, με μελέτη ποικίλων διαμερίσεων του χώρου και αποστάσεων για το distance-join ερώτημα.

8.1 ΣΥΝΟΛΑ ΔΕΔΟΜΕΝΩΝ

Ο προτεινόμενος αλγόριθμος εφαρμόστηκε σε τρία διαφορετικά ζευγάρια συνόλων δεδομένων. Όλα τα σύνολα δεδομένων είναι αρχεία της μορφής comma separated values(csv), είναι αποθηκευμένα στο HDFS και είναι της μορφής (ID,x,y), όπου ως x ορίζεται το longitude και ως y το latitude ενός σημείου, με το ID να αποτελεί το μοναδικό αναγνωριστικό χαρακτηριστικό κάθε σημείου. Αρχικά συλλέχθηκαν δεδομένα σχετικά με δημόσια κτήρια και παροχές στην χώρα της Ινδίας[19], όπου μετά την διαδικασία καθαρισμού των δεδομένων, το dataset buildings περιέχει 33.335 εγγραφές και το dataset amenities 243.124 εγγραφές. Σαν δεύτερο ζευγάρι και σαν τρίτο ζευγάρι δεδομένων, δημιουργήθηκαν ζευγάρια datasets με 1.000.000 και 2.000.000 τυχαίες εγγραφές γεωγραφικών δεδομένων το καθένα, με σκοπό την μελέτη συνδυασμού περισσότερων παρατηρήσεων.

8.2 ΣΥΓΚΡΙΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Κατά την πειραματική διαδικασία, αρχικά εκτελέστηκε ο αλγόριθμος βασιζόμενος σε καθαρή υπολογιστική ισχύ, χωρίς την διαμέριση του χώρου με την χρήση του KD-tree αλγορίθμου(Brute Force Algorithm).

Πίνακας 1. Χρόνοι Εκτέλεσης Brute Force Algorithm

Σύνολα Δεδομένων	Χρόνος Εκτέλεσης
243K X 33K	300 sec
1M X 1M	≅ 6 hours
2M X 2M	≅ 14 hours

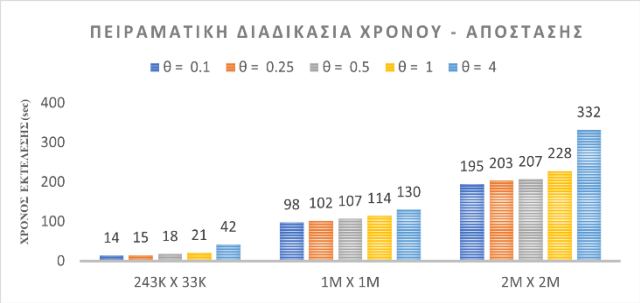
Στην συνέχεια πραγματοποιήθηκε διερεύνηση σχετικά με τις κατάλληλες διαμερίσεις στο χώρο, από τον KD-tree αλγόριθμο ανά ζευγάρι συνόλων δεδομένων με σταθερό μέτρο απόστασης θ = 0.1 για την βέλτιστη λειτουργία του αλγορίθμου. Οι χρόνοι εκτέλεσης του αλγορίθμου στο Cluster υπολογιστών παρουσιάζονται στον πίνακα 2.

Πίνακας 2. Χρόνοι Εκτέλεσης Αλγορίθμου στο Spark Cluster

Διαμερίσεις Χώρου / Σύνολα Δεδομένων	16	32	64	128
243K X 33K	18 sec	**14 sec**	30 sec	59 sec
1M X 1M	330 sec	241 sec	127 sec	**98 sec**
2M X 2M	465 sec	320 sec	235 sec	**195 sec**

Παρατηρήθηκε λοιπόν ότι η βέλτιστη διαμέριση των συνόλων δεδομένων είναι άμεσα συσχετιζόμενη με το μέγεθος των παρατηρήσεων στο οποίο θα εφαρμοστεί ο αλγόριθμος. Αναλυτικότερα, για μικρότερα σύνολα δεδομένων βέλτιστη κρίνεται η διαμέριση του χώρου σε 32 υποσύνολα, ενώ για μεγάλα σύνολα δεδομένων σε 128 υποσύνολα του χώρου. Η επιλογή αυτή προκύπτει απο το υπολογιστικό κόστος, βάση του οποίου πρέπει να αναζητείται και η βέλτιστη τομή μεταξύ διαμερίσεων του χώρου και του μεγέθους των παρατηρήσεων.

Ακολούθως με δεδομένη την βέλτιστη διαμέριση του χώρου, μελετήθηκε η εκτέλεση του αλγορίθμου με την χρήση των τριών διαφορετικών σε πλήθος εγγραφών σύνολων δεδομένων για διαφορετικές αποστάσεις θ που μπορεί να είσαγει ο χρήστης, με σκοπό την διερεύνηση της συσχέτισης του χρόνου εκτέλεσης του αλγορίθμου με την ζητούμενη απόσταση. Όπως παρουσιάζεται και στην εικόνα 3, ανάλογα και με την κατανομή-πυκνότητα των εγγραφών ανα ζευγάρι συνόλων δεδομένων, όσο αυξάνεται η απόσταση την οποία ορίζει ο χρήστης, αυξάνονται σε πλήθος και οι απαραίτητοι υπολογισμοί κάτι το οποίο επαγωγικά αυξάνει και τον χρόνο εκτέλεσης του αλγορίθμου.



Εικόνα 3. Μετρήσεις Χρόνου συναρτήσει της Απόστασης

9. ΣΥΜΠΕΡΑΣΜΑΤΑ

Με το πέρας της πειραματικής διαδικασίας και της σύγκρισης των αποτελεσμάτων, παρουσιάζεται με τον καλύτερο τρόπο η αξία των τεχνικών ευρετηρίασης-διαμέρισης του χώρου για την ταχύτερη επεξεργασία μεγάλων δεδομένων. Τα distance join ερωτήματα είναι μια από τις πιο ακριβές λειτουργίες που συνήθως υλοποιούνται στο Apache Spark, οπότε αξίζει να συρρικνώνονται με τον πιο βέλτιστο τρόπο τα δεδομένα πριν την εκτέλεσή των ερωτημάτων. Υπό αυτό το πλαίσιο, κρίνεται αναγκαία η χρήση ενός χωρικού ευρετηρίου, όπως ο KD-tree αλγόριθμος, με τον βέλτιστο αριθμό διαμερίσεων του χώρου να ορίζεται στα 128 MBRs.

Όσο αφορά τον χρόνο απόκρισης του αλγορίθμου ιδανική κρίνεται η επιλογή της οριζόντια κλιμάκωσης(scaling out), με την προσθήκη επιπλέον πόρων στο σύπλεγμα υπολογιστών που δημιουργήθηκε στα πλαίσια της εργασίας. Επιπλέον αν και στο [20] τονίζεται ότι η διαδικασία του repartition των δεδομένων είναι ιδιαίτερα ακρίβη υπολογιστικά και δεν συνιστάται σε υλοποιήσεις με μοναδική ενέργεια το ακριβό υπολογιστικά join δύο συνολών δεδομένων, δεν παρατηρήθηκε επιβράδυνση στην εκτέλεση του αλγορίθμου. Σε κάποιες περιπτώσεις μάλιστα επιτεύχθηκε και ταχύτερη εκτέλεση του αλγορίθμου στο Apache Cluster που υλοποιήθηκε, λόγω της αποφυγής του network latency κατά την διενέργεια του ελέγχου. Σίγουρα όμως, η διαδικασία του repartition στην περίπτωση διαδοχικών distance join ερωτημάτων με μικρότερες αποστάσεις θ του αρχικού ερωτήματος, επιταχύνει τον χρόνο απόκρισης του αλγορίθμου με την αποφυγή του επαναλαμβανόμενου shuffling των δεδομένων για κάθε νέο ερώτημα.

Μελλοντικά επισημαίνεται ότι ο προτεινόμενος αλγόριθμος θα μπορούσε να συνεισφέρει σε ένα σύστημα προτάσεων του οποίου οι χρήστες επιθυμούν να βρουν σημεία ενδιαφέροντος ή πληροφορίες με βάση την τοποθεσία στην οποία βρίσκονται.

10. ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Martin Skrodzki, The k-d tree data structure and a proof for neighborhood computation in expected logarithmic time, 2019

[2] Edwin H.Jacox and Hanan Samet, Online Appendix to: Spatial Join Techniques, 2007

[3] Al Aghbari, Z, et al. SparkNN: A Distributed In-Memory Data Partitioning for KNN Queries on Big Spatial Data. Data Science Journal, 19:35, 2020

[4] Daiwei Li and Haiqing Zhang, An Advanced k Nearest Neighbor Classification Algorithm Based on KD-tree, 2018

[5] Jiang Zhang, Hanqi Guo, Dynamic Load Balancing Based on Constrained K-D Tree Decomposition for Parallel Particle Tracing, 2017

[6] Randall T. Whitman, Bryan G. Marsh, Michael B. Park, and Erik G. Hoel, Distributed Spatial and Spatio-Temporal Join on Apache Spark, 2019

[7] García-García, Michael Vassilakopoulos, et al., Efficient distance join query processing in distributed spatial data management systems. Information Sciences, 2019

[8] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler, The Hadoop Distributed File System, 2010

[9] Eman Shaikh, Yasmeen Alufaisan, Iman Mohiuddin, Irum Nahvi, Apache Spark: A Big Data Processing Engine,2019

[10] Bill Chambers and Matei Zaharia, Spark: The Definitive Guide, O'Reilly Media, Inc, 2018

[11] Michael Armbrust, Reynold S. Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K. Bradley, Xiangrui Meng, Spark SQL: Relational Data Processing in Spark, 2015

[12] Jeffrey Dean, Sanjay Ghemawat Communications of the ACM, MapReduce: A Flexible Data Processing Tool, 2010

[13] Christos Doulkeridis, Akrivi Vlachou, Nikos Pelekis, Yannis Theodoridis. A Survey on Big Data Processing Frameworks for Mobility Analytics, 2021

[14] Siddiqui, Isma & Abbas, Asad & Ariffin, Abdul & Lee, Scott Uk-Jin, A Comparative Study of Multithreading APIs for Software of ICT Equipment. Indian Journal of Science and Technology, 2016

[15] Zhang, Tianyu & Lian, Xin, A dynamic re-partitioning strategy based on the distribution of key in Spark. AIP Conference Proceedings, 2018

[16] Agathangelos Giannis & Troullinou Georgia & Kondylakis Haridimos & Stefanidis Kostas & Plexousakis Dimitris. Incremental Data Partitioning of RDF Data in SPARK, 2018

[17] Vinod Kumar Vavilapallih Arun C Murthyh Chris Douglasm Sharad Agarwal, Apache Hadoop YARN: Yet Another Resource Negotiator, 2013

[18] Varnita Yadav, Vineet Sajwan, The Hadoop Distributed File System: Architecture and Internals, 2015

[19] Buildings & Amenities All Around India, https://www.kaggle.com/jyotsnasweedle/buildings-amenities-all-over-india?select=building.csv

[20] Holden Karau, Rachel Warren, High Performance Spark, Chapter 4, O'Reilly Media, Inc 2017