ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΜΣ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ & ΥΠΗΡΕΣΙΕΣ
Μεγάλα Δεδομένα & Αναλυτική

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Επεξεργασία Χωροκειμενικών Ερωτημάτων Εύρους για Μεγάλα Δεδομένα

Άγγελος Καραμπίνος

Επιβλέπων Καθηγητής: Χρήστος Δουλκερίδης

ΠΕΙΡΑΙΑΣ

Ιούνιος 2023

Περίληψη

Η αυξανόμενη χρήση του Διαδικτύου μέσω κινητών συσκευών εξοπλισμένων με GPS έχει οδηγήσει στη δημιουργία ενός τεράστιου όγκου δεδομένων χωρο-κειμενικού χαρακτήρα, τα οποία χαρακτηρίζονται από γεωγραφική θέση και κειμενική περιγραφή. Η πολυπλοκότητα στη διαχείριση των χωρο-κειμενικών δεδομένων προκύπτει από την υψηλή διαστασιμότητα του χώρου που αντιπροσωπεύουν.

Δεδομένου του μεγάλου όγκου χωρο-κειμενικών δεδομένων, έχουν αναπτυχθεί ποικίλοι τύποι ερωτημάτων για να καλύψουν διάφορες ανάγκες. Ωστόσο, αυτή η εργασία επικεντρώνεται στα χωρο-κειμενικά ερωτήματα εύρους. Ο σκοπός ενός τέτοιου ερωτήματος είναι να εντοπίζει αντικείμενα που βρίσκονται κοντά σε μια καθορισμένη τοποθεσία και επιδεικνύουν ομοιότητα με τις λέξεις-κλειδιά του ερωτήματος. Επιπλέον λόγω του όγκου τους απαιτούνται κατανεμημένα συστήματα για την επεξεργασία και αποθήκευσή τους.

Στην παρούσα διπλωματική εργασία, παρουσιάστηκε ένας αλγόριθμος ευρετηρίασης χωρο-κειμενικών δεδομένων για εκτέλεση ερωτημάτων εύρους σε κατανεμημένο περιβάλλον επεξεργασίας. Ο αλγόριθμος υλοποιήθηκε σε Apache Spark και συγκρίθηκε με spatial first και textual first μεθόδους σε Apache Spark, Apache Sedona και GeoMesa ως προς τον χρόνο εκτέλεσης.

Λέξεις Κλειδιά: Χωρο-κειμενικά δεδομένα, Χωρο-κειμενική ευρετηρίαση, Ερωτήματα εύρους, Κατανεμημένη επεξεργασία

Abstract

The widespread use of the Internet via GPS enabled mobile devices has led to the generation of an enormous volume of spatio-textual data, characterized by geographic location and textual description. The complexity of managing spatio-textual data arises from the high dimensionality of the representation space.

Given the vast volume of spatio-textual data, a variety of query types have been proposed to address diverse needs. This work, however, concentrates on spatio-textual range queries. The purpose of such a query is to identify objects that are near to a specified location and exhibit similarity with the query's keywords. Furthermore, due to the high volume of data, distributed systems become necessary for effective processing and storage.

This dissertation presents a spatio-textual indexing algorithm for conducting spatio-textual range queries within a distributed processing environment. The algorithm was implemented on Apache Spark and compared with spatial first and textual first methods in Apache Spark, Apache Sedona and GeoMesa in terms of execution time.

Keywords: Spatio-textual data, Spatio-textual index, Range queries, Distributed computing

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου κ. Δουλκερίδη Χρήστο για την ευκαιρία που μου έδωσε να ασχοληθώ με αυτό το θέμα και για την πολύτιμη καθοδήγησή του όλη την χρονία, ακόμα θα ήθελα να ευχαριστήσω την οικογένεια μου, Νίκη-Αέλια-Νεφέλη, για την συμπαράστασή και υπομονή τους καθόλη την διάρκεια του μεταπτυχιακού.

Πίνακας Περιεχομένων

1. Εισαγωγή 1
1.1 Αντικείμενο Εργασίας 2
1.2 Δομή Εργασίας 4
2. Συναφείς Ερευνητικές Εργασίες 5
2.1 Ευρετήρια Χωρο-κειμενικών Δεδομένων 5
2.2 Index Χωρο-κειμενικών Δεδομένων Για Ερωτήματα Εύρους 9
3. Τεχνολογικό Υπόβαθρο 13
3.1 Apache Spark 13
3.1.1 Αρχιτεκτονική 13
3.1.2 Βιβλιοθήκες Spark 15
3.1.3 Predicate Pushdown 16
3.2 Apache Sedona 17
3.2.1 Αρχιτεκτονική 17
3.2.2 Spatial Partitioning - Indexing 18
3.3 GeoMesa 20
3.3.1 Αρχιτεκτονική 20
3.3.2 Spatial Partitioning – Indexing 21
4. Περιγραφή Υλοποίησης 23
4.1 Κατανεμημένος Spatio-Texual Index 23
4.1.1 Pre-Processing Step 24
4.1.2 Query Step 27
5. Πειραματική Διαδικασία και Αποτελέσματα 30
5.1 Περιγραφή Πειραμάτων 30
5.2 Αποτελέσματα 33
5.2.1 Μεταβολή r 33
5.2.2 Μεταβολή t 36
5.2.3 Μεταβολή keywords 39
6. Συμπεράσματα - Μελλοντικές Επεκτάσεις 43
6.1 Συμπεράσματα 43
6.2 Μελλοντικές Επεκτάσεις 44
Βιβλιογραφία 45

Κατάλογος Εικόνων

Εικόνα 1: Δομή δεδομένων R-tree 6
Εικόνα 2: Grid based index 6
Εικόνα 3: Space-filling curve 7
Εικόνα 4: Inverted file index 8
Εικόνα 5: Bitmap index 8
Εικόνα 6: Δημιουργία κειμενικών partitions 10
Εικόνα 7: Αρχιτεκτονική Apache Spark 15
Εικόνα 8: Αρχιτεκτονική Apache Sedona 18
Εικόνα 9: Αρχιτεκτονική GeoMesa 21
Εικόνα 10: Ορθογώνια partitions στον μετασχηματισμένο χώρο 24
Εικόνα 11: Μεταβολή τιμής r - συνθετικά δεδομένα 10εκ 33
Εικόνα 12: Μεταβολή τιμής r - συνθετικά δεδομένα 20εκ 34
Εικόνα 13: Μεταβολή τιμής r - σύνολο δεδομένων Twitter 35
Εικόνα 14: Μεταβολή τιμής t - συνθετικά δεδομένα 10εκ 36
Εικόνα 15: Μεταβολή τιμής t - συνθετικά δεδομένα 20 εκ 37
Εικόνα 16: Μεταβολή τιμής t - σύνολο δεδομένων Twitter 38
Εικόνα 17: Μεταβολή των λέξεων-κλειδιών - συνθετικά δεδομένα 10 εκ 39
Εικόνα 18: Μεταβολή των λέξεων- κλειδιών - συνθετικά δεδομένα 20 εκ 40
Εικόνα 19: Μεταβολή των λέξεων- κλειδιών - σύνολο δεδομένων Twitter 41

Κατάλογος Πινάκων

Πίνακας 1: Πίνακας μεταδεδομένων spatio-textual index 27
Πίνακας 2: Ψευδοκώδικας - εκτέλεση ερωτήματος 29
Πίνακας 3: Σχήμα συνθετικών δεδομένων 30

1. Εισαγωγή

Η εκτεταμένη χρήση του διαδικτύου μέσω φορητών συσκευών που είναι εξοπλισμένες με GPS έχει δημιουργήσει μια τεράστια ποσότητα χωρο-κειμενικών δεδομένων. Τα χωρο-κειμενικά δεδομένα αναφέρονται σε πληροφορίες που συσχετίζουν συγκεκριμένες λέξεις-κλειδιά με μια γεωγραφική θέση. Πλατφόρμες κοινωνικής δικτύωσης, όπως το Twitter, το Flickr, το Instagram, το Yelp και το TripAdvisor, είναι χαρακτηριστικά παραδείγματα εφαρμογών που συλλέγουν καθημερινά χωρο-κειμενικό περιεχόμενο.

Τα χωρο-κειμενικά δεδομένα μπορούν να διακριθούν σε δύο κατηγορίες: στατικά και δυναμικά. Τα στατικά δεδομένα παραμένουν αμετάβλητα όσον αφορά τη χωρική θέση ή το περιεχόμενο του κειμένου με την πάροδο του χρόνου, όπως για παράδειγμα τα σημεία ενδιαφέροντος. Αντίθετα, τα δυναμικά δεδομένα συνεχώς εμπλουτίζονται με νέα στοιχεία, όπως τα tweets, ή υπόκεινται σε αλλαγές στη χωρική ή κειμενική τους πληροφορία, όπως για παράδειγμα τα μηνύματα πλοίων.

Καθημερινά παράγονται τεράστιοι όγκοι χωρο-κειμενικών δεδομένων. Σχεδόν όλες οι ηλεκτρονικές συναλλαγές, εκτός από την κειμενική πληροφορία, συνδέονται με ένα χωρικό ίχνος, το οποίο είναι εφικτό να ανακατασκευαστεί μέσω των συντεταγμένων GPS, των θέσεων των κεραιών κινητής τηλεφωνίας και των διευθύνσεων IP. Υπάρχει ευρύ φάσμα εφαρμογών που μπορούν να επωφεληθούν από τον μεγάλο όγκο των χωρο-κειμενικών δεδομένων που παράγονται. Μερικές από αυτές είναι:

- Η στοχευμένη διαφήμιση βάσει της τοποθεσίας, όπου οι διαφημιστικές καμπάνιες σχεδιάζονται για να φτάσουν σε χρήστες με συγκεκριμένα ενδιαφέροντα. Τυπικά, αυτές οι καμπάνιες εστιάζουν στην τοποθεσία των χρηστών, με σκοπό την αύξηση της αποτελεσματικότητας σε δυνητικούς πελάτες συγκεκριμένων γεωγραφικών περιοχών. Για παράδειγμα, εάν ένας χρήστης βρίσκεται στο χώρο ενός γυμναστηρίου και το προφίλ του παρουσιάζει ενδιαφέρον για αξεσουάρ τρεξίματος, τότε αυτός ο χρήστης θα λαμβάνει διαφημίσεις που σχετίζονται με τις αντίστοιχες καμπάνιες.

- Χαρακτηρισμός σημείων από τους χρήστες, εφαρμογές όπως το Google Maps δίνουν τη δυνατότητα σε χρήστες να καταγράφουν την τοποθεσία τους και να τη συνδέουν με κειμενικές περιγραφές σχετικές με διάφορα γεγονότα, όπως ατυχήματα ή κλειστούς δρόμους. Οι υπόλοιποι χρήστες λαμβάνουν ειδοποιήσεις σε πραγματικό χρόνο σχετικά με αυτά τα γεγονότα.
- Η χωρο-κειμενική αναζήτηση στο διαδίκτυο, ο τεράστιος όγκος χωρο-κειμενικών δεδομένων, έχει οδηγήσει στη δημιουργία μηχανών αναζήτησης που είναι location-aware. Αυτό σημαίνει ότι λαμβάνουν υπόψη τόσο τη χωρική όσο και την κειμενική πληροφορία για να βελτιώσουν τη συνάφεια των αποτελεσμάτων τους.

Τα χωρο-κειμενικά δεδομένα παρουσιάζουν σημαντικές προκλήσεις στη διαχείρισή τους, εξαιτίας της υψηλής διαστασιμότητας του χώρου που αναπαριστώνται. Η ανάγκη για αποτελεσματικές μεθόδους αναζήτησης σε τέτοιου είδους δεδομένα έχει προσελκύσει μεγάλο ενδιαφέρον από την ερευνητική κοινότητα τα τελευταία χρόνια [1].

1.1 Αντικείμενο Εργασίας

Τα χωρο-κειμενικά ερωτήματα αποτελούν επεκτάσεις των χωρικών ερωτημάτων με σκοπό να συμπεριλάβουν στην αναζήτηση και την κειμενική πληροφορία των αντικειμένων. Οι πιο γνωστές κατηγορίες χωρο-κειμενικών ερωτημάτων είναι οι παρακάτω:

- Boolean kNN query: Δεδομένου ενός σημείου q, το ερώτημα στοχεύει στην εύρεση των k πλησιέστερων αντικειμένων, τα οποία περιέχουν όλες τις λέξεις-κλειδιά του ερωτήματος. Τα αποτελέσματα ταξινομούνται βάση της απόστασής τους από το q.
- Top-k kNN query: Το ερώτημα επιστρέφει τα k αντικείμενα με το υψηλότερο σκορ, το οποίο υπολογίζεται ως συνδυασμός της απόστασης του αντικειμένου από το σημείο q και της ομοιότητας των λέξεων-κλειδιών του ερωτήματος με την κειμενική περιγραφή του αντικειμένου.

- Boolean range query: Το ερώτημα στοχεύει στην ανάκτηση όλων των αντικειμένων που βρίσκονται σε απόσταση r από το q και περιέχουν όλες τις λέξεις-κλειδιά του ερωτήματος.

Η εργασία ασχολείται με την εκτέλεση χωρο-κειμενικών ερωτημάτων εύρους (range queries) σε κατανεμημένο περιβάλλον επεξεργασίας. Δεδομένου ενός ερωτήματος q που αποτελείται από μια τοποθεσία (q.x, q.y) και ενός συνόλου λέξεων-κλειδιών Q, ο σκοπός ενός ερωτήματος εύρους είναι να ανακτηθούν όλα τα χωρο-κειμενικά αντικείμενα που βρίσκονται σε απόσταση r από το q και η ομοιότητα των λέξεων-κλειδιών τους με το Q είναι μεγαλύτερη από την τιμή t. Όπου τα r και t ορίζονται από τον χρήστη και μπορεί να διαφέρουν από ερώτημα σε ερώτημα.

Τα χωρο-κειμενικά ερωτήματα εύρους με τα οποία ασχολείται η εργασία, αποτελούν επέκταση των boolean range queries. Σε αντίθεση με το boolean range query, αυτά τα ερωτήματα δεν ελέγχουν απλώς αν τα αποτελέσματα περιέχουν όλες τις λέξεις-κλειδιά του ερωτήματος, αλλά εισάγουν την έννοια της κειμενικής ομοιότητας. Αυτό σημαίνει ότι τα αποτελέσματα ταξινομούνται και επιλέγονται με βάση μια τιμή κατωφλίου, η οποία αντιπροσωπεύει το ελάχιστο επίπεδο κειμενικής ομοιότητας που απαιτείται. Αυτή η προσέγγιση προσφέρει μεγαλύτερη ευελιξία σε σχέση με τα boolean range queries, τα οποία ανακτούν αντικείμενα με λέξεις-κλειδιά που απλώς ταυτίζονται με αυτές του ερωτήματος.

Η εκτέλεση χωρο-κειμενικών ερωτημάτων εύρους έχει πολυπλοκότητα λόγω της υψηλής διαστασιμότητας του χώρου των δεδομένων. Στην εργασία [2] προτείνεται ένας αλγόριθμος που μετασχηματίζει έναν πολυδιάστατο χώρο σε δισδιάστατο με σκοπό την αποτελεσματική εκτέλεση χωρο-κειμενικών ερωτημάτων εύρους. Η παρούσα εργασία επιχειρεί να διερευνήσει την εφαρμογή αυτού του αλγορίθμου στο πλαίσιο κατανεμημένων χωρο-κειμενικών ερωτημάτων εύρους. Μία από τις προκλήσεις στην κατανεμημένη επεξεργασία δεδομένων είναι ο αποτελεσματικός και δικαιος καταμερισμός των δεδομένων μεταξύ των μονάδων επεξεργασίας. Αυτό επιτρέπει την αξιοποίηση των παράλληλων υπολογιστικών πόρων, δίνοντας τη δυνατότητα για επεκτασιμότητα, δηλαδή τη μείωση του χρόνου εκτέλεσης των ερωτημάτων καθώς αυξάνεται ο αριθμός των μονάδων επεξεργασίας.

Στην παρούσα εργασία παρουσιάζεται ένας αλγόριθμος ευρετηρίασης για την κατανεμημένη επεξεργασία χωρο-κειμενικών ερωτημάτων εύρους. Ο αλγόριθμος έχει υλοποιηθεί στο κατανεμημένο περιβάλλον επεξεργασίας Apache Spark και έχει συγκριθεί με άλλους αλγορίθμους που έχουν προταθεί για την επεξεργασία χωρο-κειμενικών δεδομένων και υλοποιήθηκαν επίσης στο Spark, καθώς και σε επεκτάσεις του που υποστηρίζουν χωρικά δεδομένα.

1.2 Δομή Εργασίας

Η δομή της εργασίας διαμορφώνεται ως εξής:

- Στο κεφάλαιο 2, παρουσιάζονται συναφείς ερευνητικές εργασίες που ασχολούνται με την κατασκευή ευρετήριων πάνω σε χωρο-κειμενικά δεδομένα για την εκτέλεση ερωτημάτων σε αυτά.
- Στο κεφάλαιο 3, γίνεται μια σύντομη αναφορά σχετικά με το τεχνολογικό υπόβαθρο που απαιτείται για την κατανόηση της εργασίας.
- Στο κεφάλαιο 4, παρουσιάζεται ο αλγόριθμος που προτείνεται στα πλαίσια της εργασίας για την εκτέλεση χωρο-κειμενικών ερωτημάτων εύρους σε κατανεμημένο περιβάλλον επεξεργασίας.
- Στο κεφάλαιο 5, περιέχεται η περιγραφή της πειραματικής διαδικασίας που ακολουθήθηκε, οι μέθοδοι ανταγωνιστές που θα συγκριθούν με τον προτεινόμενο αλγόριθμο καθώς και τα αποτελέσματα των πειραμάτων.
- Στο κεφάλαιο 6, ολοκληρώνεται η εργασία με τα συμπεράσματα και προτάσεις για μελλοντικές επεκτάσεις.

2. Συναφείς Ερευνητικές Εργασίες

Σε αυτό το κεφάλαιο παρουσιάζονται δημοσιεύσεις που σχετίζονται με την παρούσα εργασία.

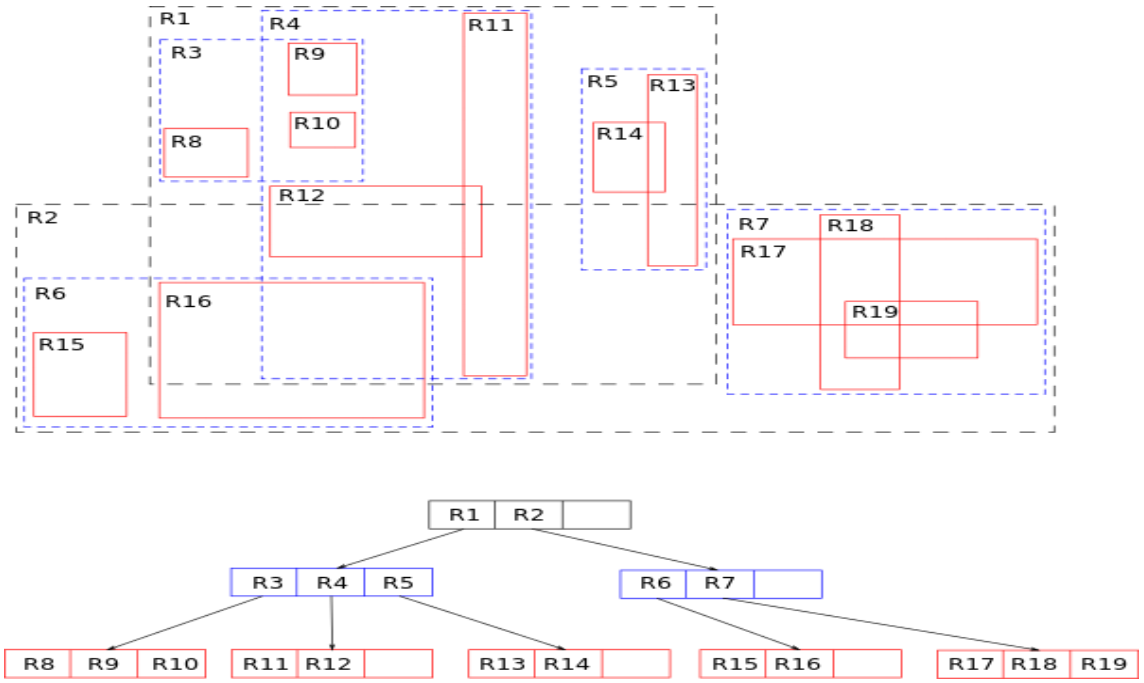
2.1 Ευρετήρια Χωρο-κειμενικών Δεδομένων

Η ευρετηρίαση χωρο-κειμενικών δεδομένων είναι ένα ενεργό πεδίο έρευνας με πολλές σημαντικές εργασίες. Η ευρετηρίαση αυτή ενσωματώνει δύο βασικά στοιχεία, την ευρετηρίαση της κειμενικής πληροφορίας και την ευρετηρίαση της χωρικής, με σκοπό την εκτέλεση ερωτημάτων που συνδυάζουν αυτά τα δύο στοχεία.

Έχουν προταθεί διαφορετικοί indexes για τη διαχείριση χωρο-κειμενικών δεδομένων. Οι indexes αυτοί αποτελούνται από μια δομή χωρικού ευρετηρίου και μια δομή ευρετηρίου κειμένου.

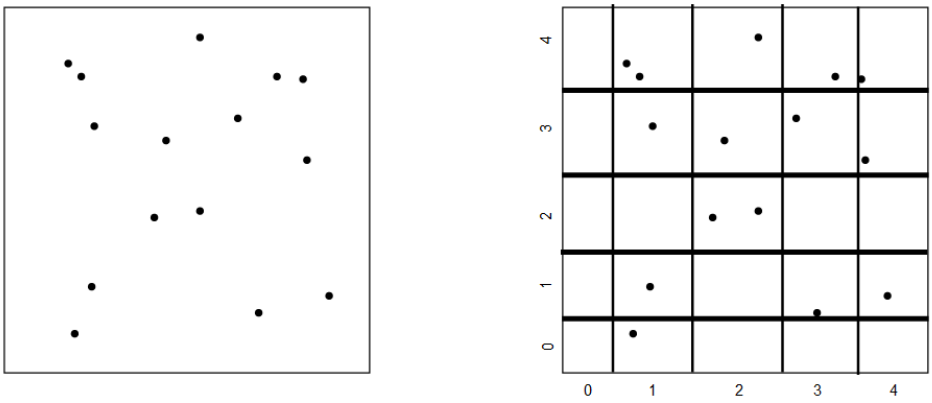
Ως προς το χωρικό τους ευρετήριο διακρίνονται στις εξής κατηγορίες:

R-tree based: Το R-tree αποτελεί μια ιεραρχική δομή δεδομένων η οποία χρησιμοποιείται για την ευρετηριαση πολυδιάστατων αντικειμένων. Η βασική ιδέα είναι η ομαδοποίηση συναφών αντικειμένων, τα οποία στη συνέχεια απεικονίζονται μέσω του ελάχιστου οριοθετημένου ορθογωνίου (Minimum Bounding Rectangle - MBR). Κάθε κόμβος στο R-tree αντοιστοιχεί στο MBR που καλύπτει τα παιδία του ενώ στα φύλλα του δέντρου περιλαμβάνουν τα αντικείμενα [4, 7, 8, 9, 10, 13, 14].



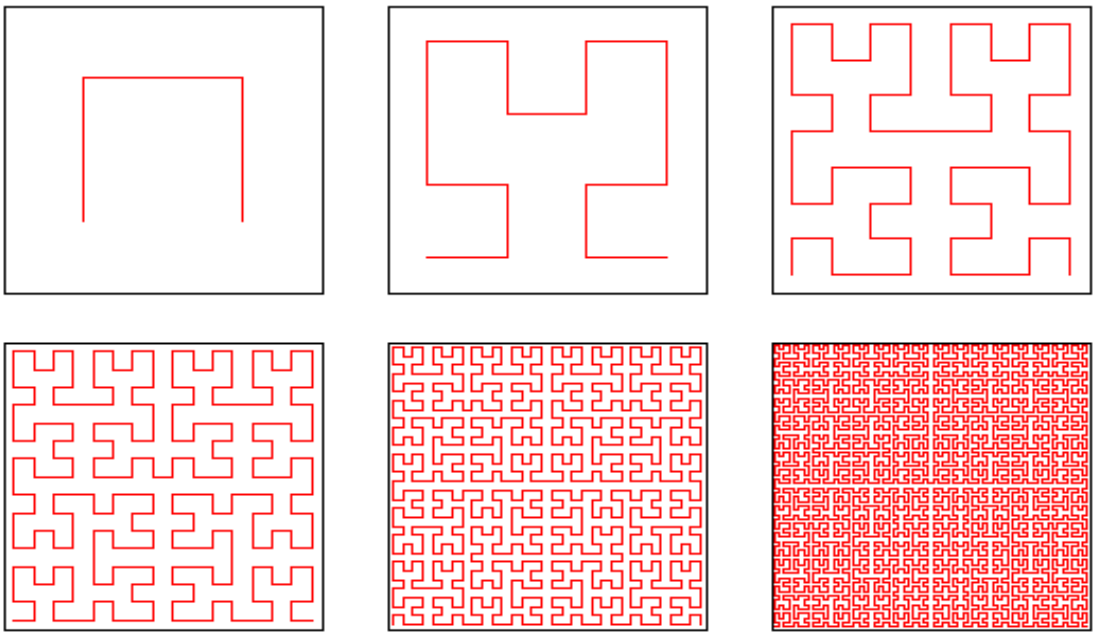
Εικόνα 1: Δομή δεδομένων R-tree (source)

Grid-based: Αυτή η προσέγγιση χωρίζει τον χώρο σε ένα πλέγμα κελιών και τα αντικείμενα χαρτογραφούνται σε αυτά τα κελιά με βάση τη χωρική τοποθεσία τους. Αυτό προσφέρει μια απλή και αποτελεσματική λύση για την αναζήτηση χωρικών δεδομένων [11, 12].



Εικόνα 2: Grid based index

Space-filling curve based: Οι καμπύλες πλήρωσης χώρου αποτελούν έναν ειδικό τύπο μαθηματικής συνάρτησης που χρησιμοποιείται στην ευρετηρίαση χωρικών δεδομένων. Κατά την εφαρμογή αυτής της συνάρτησης, κάθε σημείο σε ένα πολυδιάστατο χώρο μετατρέπεται σε ένα σημείο στην καμπύλη. Ένα από τα κύρια χαρακτηριστικά της μεθόδου αυτής είναι ότι τα κοντινά σημεία στον αρχικό χώρο παραμένουν κοντινά και στην καμπύλη. Οι πιό γνωστές τεχνικές ευρετηρίασης χωρικών δεδομένων που βασίζονται σε αυτή την τεχνική είναι η καμπύλη Hilbert και η καμπύλη Z [5, 6].

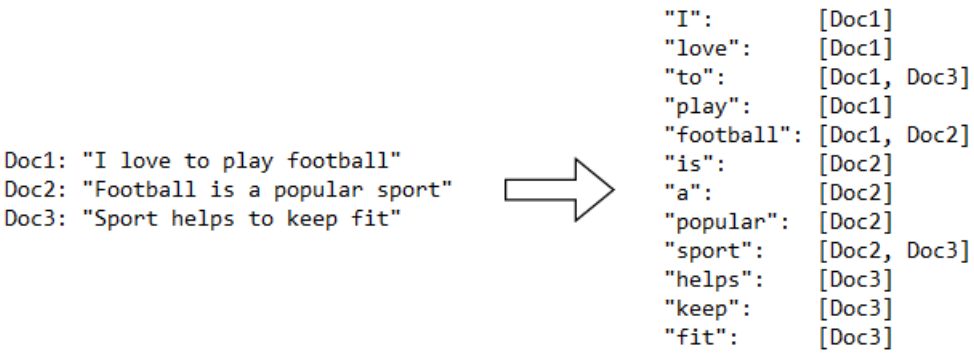


Εικόνα 3: Space-filling curve (source)

Ως προς το κειμενικό ευρετήριο διακρίνονται στις παρακάτω κατηγορίες:

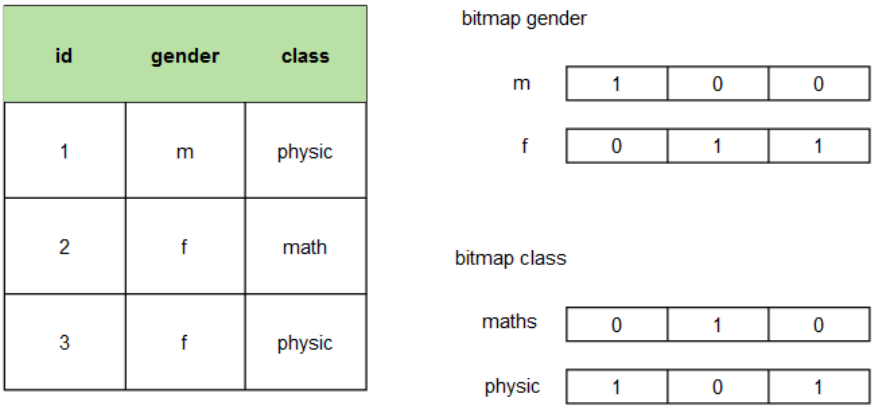
Inverted file index: Ένα ανεστραμμένο ευρετήριο είναι μια δομή δεδομένων που χρησιμοποιείται για τη δημιουργία ενός ευρετηρίου με δυνατότητα αναζήτησης δεδομένων που βασίζονται σε κείμενο. Ο τρόπος με τον οποίο λειτουργεί ένα ανεστραμμένο ευρετήριο είναι η αποθήκευση μιας λίστας αναφορών για κάθε λέξη, που περιέχει τα έγγραφα στα οποία περιέχεται η λέξη. Συνεπώς, το ανεστραμμένο ευρετήριο αντιστρέφει την τυπική σχέση μεταξύ εγγράφων και λέξεων, αντί να καταγράφει τις λέξεις που εμφανίζονται σε ένα

έγγραφο, καταγράφει τα έγγραφα στα οποία μια λέξη εμφανίζεται [5, 6, 7, 9, 10, 11, 12, 14].



Εικόνα 4: Inverted file index

Bitmap index: Το bitmap είναι ένα είδος ευρετηρίασης που χρησιμοποιεί πίνακες από bits για να απαντήσει σε ερωτήματα. Η βασική ιδέα πίσω από ένα ευρετήριο bitmap είναι ότι για κάθε μοναδική τιμή στη στήλη που ευρετηριάζει, δημιουργεί ένα πίνακα από bits, όπου κάθε bit αναπαριστά μια γραμμή στον πίνακα. Αν η τιμή της στήλης για μια συγκεκριμένη γραμμή αντιστοιχεί στην τιμή που ευρετηριάζουμε, θέτουμε το αντίστοιχο bit στο bitmap σε 1. Διαφορετικά, το ορίζουμε σε 0. Αυτή η τεχνική ευρετηρίασης προσφέρει γρήγορη αναζήτηση και αποτελεσματική επεξεργασία ερωτημάτων που συνδυάζουν πολλούς όρους [4, 8, 13].



Εικόνα 5: Bitmap index

Πολλοί από αυτούς τους indexes προτείνουν μια spatial first ή textual first προσέγγιση, ενώ άλλοι προτείνουν μια υβριδική, η οποία συνδυάζει χαρακτηριστικά από το χωρικό και από το κειμενικό ευρετήριο, με τέτοιο τρόπο ώστε και οι δύο πληροφορίες να χρησιμοποιηθούν κατά την αναζήτηση ενός αντικειμένου στον χώρο.

Συγκεκριμένα, στην [7] το χωρικό ευρετήτιο R-tree έχει επεκταθεί ώστε κάθε εσωτερικός κόμβος να περιέχει τις λέξεις-κλειδία που αντιστοιχούν στα ανικείμενα τα οποία περιλαμβάνονται στο υποδέντρο με ρίζα τον αντίστοιχο κόμβο. Ενώ στην [6] το ανεστραμένο ευρετήριο επεκτείνεται ώστε κάθε λέξη να περιέχει έναν πίνακα με την χωρική πληροφορία των αντικειμένων που αντιπροσωπέυει.

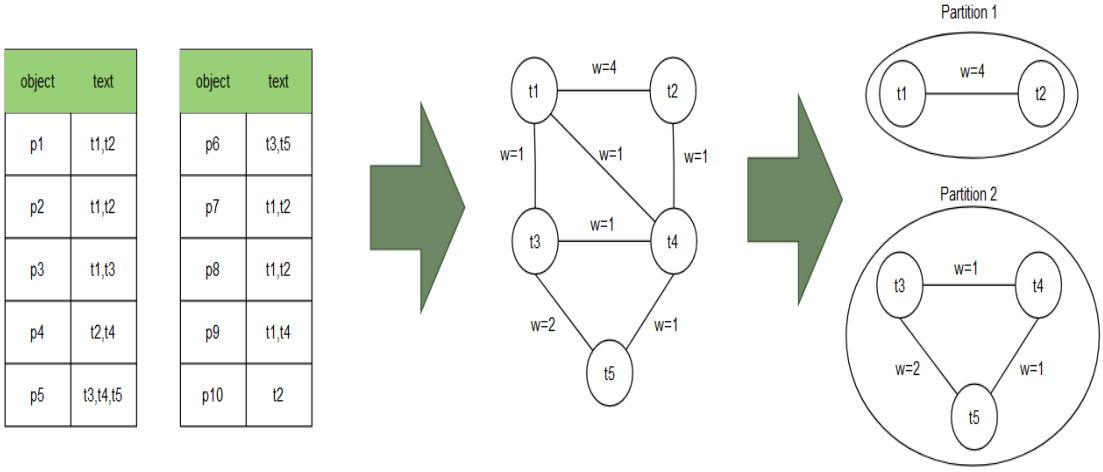
2.2 Index Χωρο-κειμενικών Δεδομένων Για Ερωτήματα Εύρους

Στην εργασία [2] προτείνεται ένας αλγόριθμος ευρετηρίασης χωρο-κειμενικών δεδομένων για την αποτελεσματική εκτέλεση χωρο-κειμενικών ερωτημάτων εύρους. Η μέθοδος που προτείνεται για τον σκοπό αυτό, αφορά τη μετατροπή ενός πολυδιάστατου χώρου σε δισδιάστατο, όπου η μία διάσταση αναπαριστά τη χωρική απόσταση και η άλλη την κειμενική ομοιότητα.

Στον μετασχηματισμένο χώρο δημιουργούνται διαμερίσεις (partitions), όπου η καθεμία περιέχει δεδομένα με κοντινή απόσταση μεταξύ τους και με κοινές λέξεις-κλειδιά. Ο αλγόριθμος βασίζεται στο iDistance [3], μια μέθοδο ευρετηρίασης που χρησιμοποιεί την απόσταση των σημείων μεταξύ τους και όχι την θέση τους.

Όσον αφορά το μετασχηματισμό στη χωρική πληροφορία, τα δεδομένα στον αρχικό χώρο ομαδοποιούνται σε clusters μέσω ενός κατάλληλου αλγορίθμου ομαδοποίησης. Στη συνέχεια, για κάθε αντικείμενο p υπολογίζεται η iDistance τιμή του μέσω της συνάρτησης: $iDist(p) = i * c + dist(K_i, p)$ όπου το: K_i είναι το centroid του cluster C_i στο οποίο ανήκει το p, και το c αποτελεί μια σταθερά αρκετά μεγάλη ούτως ώστε όλα τα αντικείμενα από το cluster C_i να μετασχηματίζονται στο διάστημα $[i * c, (i + 1) * c]$.

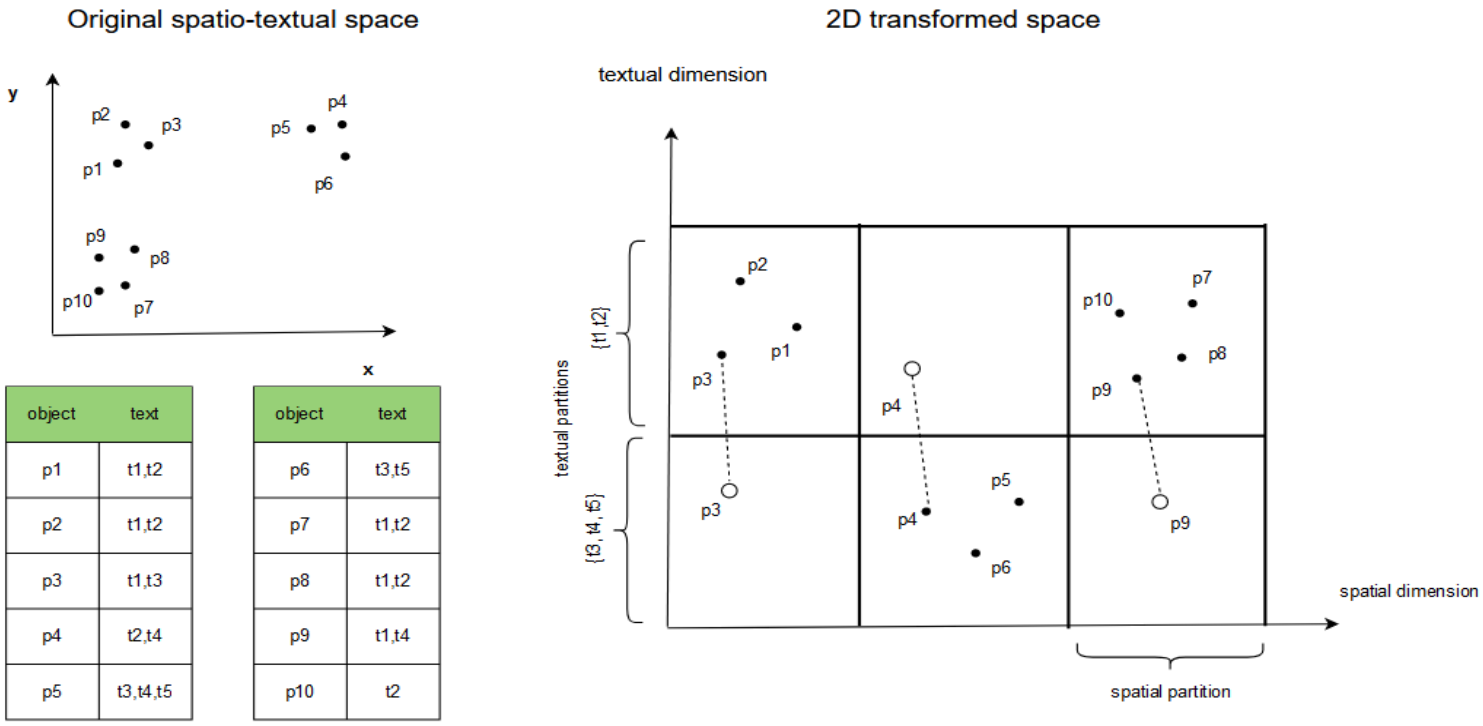
Για τον μετασχηματισμό στη κειμενική πληροφορία, ο στόχος είναι να δημιουργηθούν k σύνολα λέξεων-κλειδιών ξένα μεταξύ τους, όπου η ένωσή τους αποτελεί το λεξιλόγιο του συνόλου δεδομένων. Για τον σκοπό αυτό, αρχικά δημιουργείται ένας μη κατευθυνόμενος γράφος με βάρη, όπου για κορυφές έχει τις λέξεις-κλειδιά. Μια ακμή μεταξύ δύο κορυφών υπάρχει, εφόσον αυτές οι λέξεις-κλειδιά συνυπάρχουν σε κάποιο αντικείμενο. Το βάρος κάθε ακμής ισούται με το πλήθος των αντικειμένων που συνυπάρχουν οι αντίστοιχες λέξεις-κλειδιά. Στην συνέχεια, εκτελείται ένας αλγόριθμος κατάτμησης σε γράφους με αποτέλεσμα να δημιουργούνται k σύνολα με λέξεις-κλειδιά ξένα μεταξύ τους, τέτοια ώστε οι λέξεις που συνυπάρχουν συχνά, να εντάσσονται στο ίδιο σύνολο.



Εικόνα 6: Δημιουργία κειμενικών partitions (source)

Τα αντικείμενα στον αρχικό χώρο που περιέχουν λέξεις-κλειδιά από περισσότερα από ένα σύνολα, αντιγράφονται στον μετασχηματισμένο χώρο τόσες φορές όσες και ο αριθμός των συνόλων που ανήκουν οι λέξεις τους. Στη συνέχεια, σε αντιστοιχία με το χωρικό κομμάτι υπολογίζεται η τιμή που θα αντιστοιχηθεί η κειμενική πληροφορία μέσω της συνάρτησης: $iSim(p, V_i) = i * c' + \frac{|P \cap V_i|}{P}$ όπου: το V_i είναι το υποσύνολο με τις λέξεις-κλειδιά που περιέχει το αντικείμενο p, το P είναι το σύνολο των λέξεων-κλειδιών του αντικείμενου p και το c' είναι μια σταθερά αρκετά μεγάλη ούτως ώστε όλα τα αντικείμενα από το cluster V_i να μετασχηματίζονται στο διάστημα $[i * c', (i + 1) * c']$.

Η βασική διαφορά της τεχνικής μετασχηματισμού της κειμενικής πληροφορίας και της μεθόδου iDistance, που χρησιμοποιήθηκε για τον μετασχηματισμό της χωρικής πληροφορίας, είναι ότι κάθε αντικείμενο μπορεί να ανατεθεί σε περισσότερα από ένα σύνολα V_i, αναλόγως με τις λέξεις-κλειδιά που περιέχει. Στην τεχνική iDistance, κάθε αντικείμενο ανατίθεται αποκλειστικά σε ένα συγκεκριμένο χωρικό cluster.



object	text
p1	t1,t2
p2	t1,t2
p3	t1,t3
p4	t2,t4
p5	t3,t4,t5

object	text
p6	t3,t5
p7	t1,t2
p8	t1,t2
p9	t1,t4
p10	t2

Εικόνα 7: Αρχικός και μετασχηματισμένος χώρος (source)

Στην παραπάνω εικόνα, παρουσιάζεται η αναπαράσταση της χωρικής και κειμενικής πληροφορίας ενός χωρο-κειμενικού συνόλου δεδομένων και ο τρόπος μετασχηματισμού τους με βάση τον αλγόριθμο που αναλύθηκε προηγουμένως. Παρατηρούμε ότι στον αρχικό χώρο διαμορφώνονται τρία χωρικά clusters, επομένως, στον μετασχηματισμένο χώρο, στον άξονα x δημιουργούνται τρία χωρικά partitions.

Σχετικά με το χωρικό κομμάτι, τα κειμενικά clusters, μετά την δημιουργία του γράφου και την εκτέλεση του αλγορίθμου κατάτμησης, αποτελούνται από δύο

σύνολα, $\{t1,\ t2\}$ και $\{t3,\ t4,\ t5\}$. Επομένως, στον μετασχηματισμένο χώρο δημιουργούνται δύο κειμενικά partitions στον άξονα y.

Αρχικά, μετασχηματίζεται η χωρική πληροφορία, κατά την οποία διεξάγεται ο υπολογισμός της τιμής idistance για τα αντικείμενα p1 έως p10, βάση της συνάρτησης: $iDist(p) = i * c + dist(K_i, p)$, που αντιστοιχεί στην τιμή του άξονα x του μετασχηματισμένου χώρου. Στη συνέχεια, υπολογίζεται η τιμή idistance για την κειμενική πληροφορία, βάση της συνάρτησης $iSim(p, V_i) = i * c' + \frac{|P \cap V_i|}{P}$ η οποία αντιστοιχεί στην τιμή του άξονα y του μετασχηματισμένου χώρου. Είναι σημαντικό να σημειωθεί ότι τα αντικείμενα p1, p2, p3 περιέχουν λέξεις-κλειδιά που ανήκουν και στα δύο κειμενικά clusters. Για τον λόγο αυτό, τα αντικείμενα αυτά μετασχηματίζονται και στα δύο κειμενικά partitions του μετασχηματισμένου χώρου, δηλαδή στο {t1,t2} και στο {t3,t4,t5}.

3. Τεχνολογικό Υπόβαθρο

Σε αυτό το κεφάλαιο παρουσιάζονται οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της εργασίας.

3.1 Apache Spark

Το Apache Spark αποτελεί ένα σύστημα ανοιχτού κώδικα γενικής χρήσης, κατανεμημένης επεξεργασίας, το οποίο έχει σχεδιαστεί ειδικά για την επεξεργασία και την ανάλυση μεγάλων όγκων δεδομένων. Έχει αναπτυχθεί από τον Matei Zaharia το 2009, ενώ ήταν φοιτητής στο πανεπιστήμιο UC Berkeley. Έχει υλοποιηθεί σε Scala και προσφέρει διεπαφές σε Java, Python και R. Τα βασικά χαρακτηριστικά του είναι η ανθεκτικότητά σε σφάλματα και η ικανότητά για αποτελεσματικό παραλληλισμό δεδομένων [16, 17].

3.1.1 Αρχιτεκτονική

Τα βασικά συστατικά στοιχεία του Spark είναι ο driver, οι executors και ο cluster manager. Ο driver εκτελεί τη μέθοδο main του προγράμματος και είναι υπεύθυνος για τη διατήρηση πληροφοριών σχετικά με την εφαρμογή, την ανταπόκριση στην είσοδο του χρήστη και τον προγραμματισμό των εργασιών στους executors. Επιπλέον, ο driver επικοινωνεί με τον cluster manager για να έχει πρόσβαση στους πόρους του cluster. Κάθε executor είναι υπεύθυνος για την εκτέλεση του κώδικα που του έχει ανατεθεί από τον driver και για την αναφορά της κατάστασης των υπολογισμών πίσω σε αυτόν.

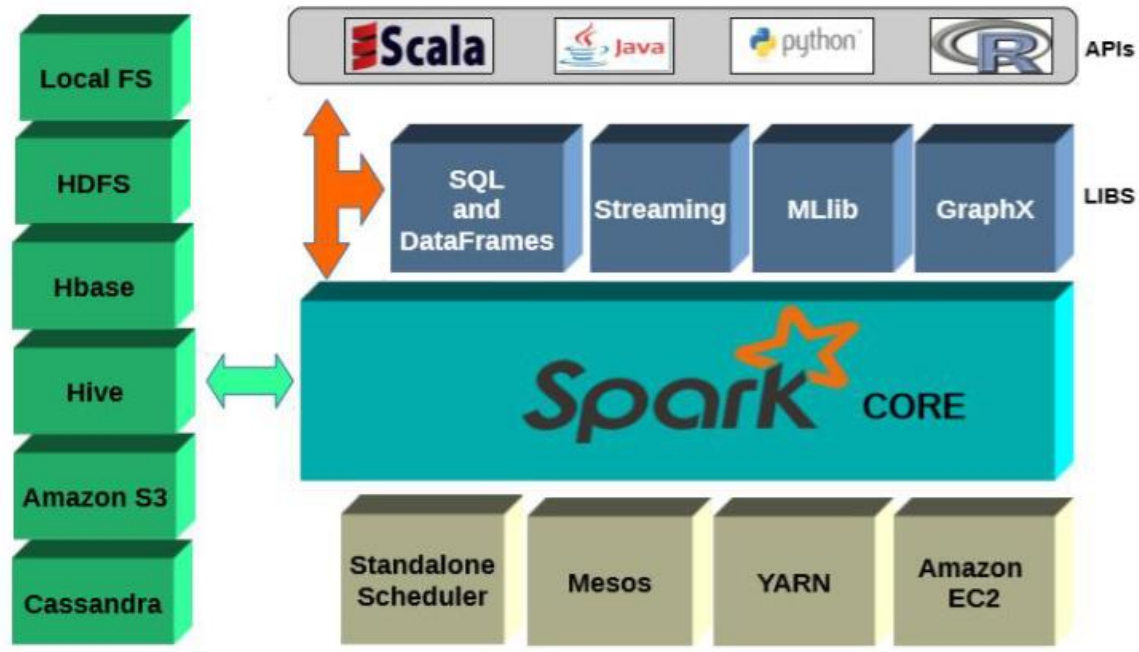
Ένα από τα βασικά χαρακτηριστικά του Spark είναι η ικανότητά να εκτελεί επεξεργασία δεδομένων στη μνήμη. Το Spark διατηρεί τα δεδομένα στη μνήμη όσο το δυνατόν περισσότερο, για να αποφεύγει το αργό διάβασμα από τον δίσκο. Αυτό το χαρακτηριστικό του, το διακρίνει από τα υπόλοιπα συστήματα επεξεργασίας μεγάλων δεδομένων, όπως το MapReduce, τα οποία χρησιμοποιούν πολύ I/O του δίσκου.

Για να επιτρέψει το Spark την παράλληλη επεξεργασία σε κάθε executor, χωρίζει τα δεδομένα σε ενότητες που ονομάζονται partitions. Κάθε partition αποτελεί μια συλλογή δεδομένων που βρίσκεται σε έναν κόμβο του cluster. Ο αριθμός των partitions και των executors συνδέεται άμεσα με την παράλληλη επεξεργασία του Spark, καθώς κάθε executor είναι υπεύθυνος για την εκτέλεση των οδηγιών που του έχουν ανατεθεί από τον driver σε έναν συγκεκριμένο αριθμό από partitions. Η αύξηση του αριθμού των executors συμβάλλει στην αυξημένη παράλληλη επεξεργασία των partitions, ενισχύοντας έτσι την αποδοτικότητα του Spark.

Στο Spark υπάρχουν δύο βασικές κατηγορίες λειτουργιών, οι μετασχηματισμοί (transformations) και οι ενέργειες (actions). Οι δομές δεδομένων στο Spark είναι αμετάβλητες (immutables), συνεπώς δεν επιτρέπεται η τροποποίησή τους μετά τη δημιουργία τους. Μέσω των μετασχηματισμών, ο χρήστης μπορεί να ορίσει τον τρόπο με τον οποίο θα τροποποιηθεί μια δομή δεδομένων. Όλοι οι μετασχηματισμοί που εφαρμόζονται σε μια δομή δεδομένων δεν εκτελούνται άμεσα, αλλά αποθηκεύονται σε ένα γράφημα υπολογισμού οδηγιών και εκτελούνται μόλις εφαρμοστεί κάποια ενέργεια. Αυτή η τεχνική είναι γνωστή ως lazy evaluation και προσφέρει σημαντικά οφέλη στην απόδοση, καθώς το Spark μπορεί να βελτιστοποιήσει το πλάνο εκτέλεσης.

Υπάρχουν δύο ειδών μετασχηματισμοί: οι narrow dependencies και οι wide dependencies. Στους narrow μετασχηματισμούς, κάθε partition εισόδου συνεισφέρει μόνο σε ένα partition εξόδου. Συναρτήσεις αυτού του είδους είναι η map και η filter. Στους wide μετασχηματισμούς, το κάθε partition εισόδου συνεισφέρει σε πολλαπλά partition εξόδου. Συναρτήσεις αυτού του είδος είναι η groupByKey και η reduceByKey.

Οι ενέργειες είναι αυτές που θα ενεργοποιήσουν τους υπολογισμούς από μια σειρά από μετασχηματισμούς και θα επιστρέψουν ένα αποτέλεσμα. Διακρίνονται σε τρεις κατηγορίες: ενέργειες για εμφάνιση δεδομένων στην κονσόλα, ενέργειες για συλλογή δεδομένων σε αντικείμενα και ενέργειες για εγγραφή δεδομένων σε πηγές εξόδου.



Εικόνα 7: Αρχιτεκτονική Apache Spark

3.1.2 Βιβλιοθήκες Spark

Το Spark ενσωματώνει μια πληθώρα βιβλιοθηκών που επεκτείνουν τις λειτουργικές του δυνατότητες. Ενδεικτικά, αναφέρονται οι εξής:

- **Spark SQL**: Το Spark SQL είναι μια βιβλιοθήκη του Spark που έχει ως βάση το Shark [23]. Δίνει τη δυνατότητα για εκτέλεση SQL ερωτημάτων σε δομημένα και ημι-δομημένα δεδομένα και υποστηρίζει πολλαπλές μορφές δεδομένων, όπως JSON, Parquet και Hive.
- **Structured Streaming**: Το Structured Streaming παρέχει τη δυνατότητα επεξεργασίας σε πραγματικό χρόνο δεδομένων σε συνεχή ροή. Τα δεδομένα μπορεί να προέρχονται από διαφορετικές πηγές όπως, Flume, Apache Kafka, log files, web κ.α. Προσφέρει υψηλό βαθμό ανοχής σφαλμάτων και κλιμάκωσης.
- **MLlib:** Η βιβλιοθήκη MLlib παρέχει στο Spark βασικές λειτουργίες μηχανικής μάθησης. Περιέχει μια συλλογή υλοποιημένων αλγορίθμων μηχανικής μάθησης σχετικά με την ταξινόμηση, παλινδρόμηση, συσταδοποίηση και συστάσεων (recommendations). Διαθέτει επίσης μια ευρεία γκάμα εργαλείων και χαρακτηριστικών, όπως τη μείωση

διαστάσεων, τη δημιουργία σωληνώσεων (pipelines) για τη μηχανική μάθηση, γραμμική άλγεβρα και πολλά άλλα.

- **GraphX**: Το GraphX είναι η βιβλιοθήκη που χειρίζεται γράφους και υποστηρίζει τα RDG (Resilient Distributed Graph). Τα RDG αποτελούν επέκταση των RDDs και συσχετίζουν τις εγγραφές με τις κορυφές και τις ακμές ενός γράφου. Παρέχει ποικιλία τελεστών για την επεξεργασία γράφων, όπως subgraph, και υποστηρίζει τεχνικές όπως το PageRank [24] και το Pregel [25].
- **SparkR**: Το SparkR είναι η βιβλιοθήκη που παρέχει τα απαραίτητα APIs για την επεξεργασία δεδομένων μέσω της γλώσσας R. Επιτρέπει στον χρήστη να γράφει κώδικα σε R και να εκτελείται σε Spark cluster.

3.1.3 Predicate Pushdown

Το Apache Spark ακολουθεί τη λογική του declarative programming, όπου ο χρήστης προγραμματίζει τις επιθυμητές λειτουργίες σε μια διεπαφή υψηλή επιπέδου. Ο τρόπος με τον οποίο οι λειτουργίες αυτές εκτελούνται καθορίζεται από εσωτερικούς μηχανισμούς του Spark, οι οποίοι δημιουργούν ένα πλάνο εκτέλεσης που βελτιστοποιείται μέσω ενός συστατικού στοιχείου του Spark που ονομάζεται Catalyst Optimizer. Το βελτιστοποιημένο αυτό πλάνο μεταφράζεται σε γλώσσα μηχανής (JVM bytecode) και εκτελείται στους executors.

Μια από τις βελτιστοποιήσεις που εφαρμόζεται σε ένα πλάνο εκτέλεσης είναι το predicate pushdown. Κατά την εκτέλεση ενός πλάνου, όταν εφαρμόζεται ένας μετασχηματισμός φίλτρου, ο Catalyst Optimizer ελέγχει τη δυνατότητα μεταφοράς αυτού του φίλτρου στο επίπεδο της πηγής των δεδομένων. Αυτή η μορφή φίλτρου εφαρμόζεται πριν τη φόρτωση των δεδομένων στη μνήμη του Spark, επιφέροντας μείωση του I/O και της ποσότητας των δεδομένων που θα πρέπει να επεξεργαστεί το Spark, επιτυγχάνοντας έτσι σημαντική βελτίωση της απόδοσης.

Υπάρχουν δύο διαφορετικά είδη predicate pushdown που υποστηρίζονται από το Spark, τα pushed filters και τα partition filters. Τα partition filters απαιτούν την παρουσία δεδομένων που είναι partitioned βάσει ενός συγκεκριμένου χαρακτηριστικού, με τον predicate του φίλτρου να χρησιμοποιεί αυτό το

χαρακτηριστικό. Από την άλλη, τα pushed filters χρησιμοποιούν τα μεταδεδομένα των αρχείων που διαβάζονται. Κατά συνέπεια, η προϋπόθεση είναι ότι η μορφή των αρχείων υποστηρίζει μεταδεδομένα. Οι μορφές αρχείων που υποστηρίζουν τα pushed filters είναι το Parquet και το ORC. Και οι δύο αυτές μορφές αποθηκεύουν τις εγγραφές ανά στήλη και κρατάνε μεταδεδομένα για κάθε στήλη, όπως η μέγιστη και η ελάχιστη τιμή της. Επιπλέον, τα pushed filters υποστηρίζονται μέσω ενός JDBC connector όταν το Spark ανακτά δεδομένα από μια βάση δεδομένων. Σε αυτήν την περίπτωση, τα φίλτρα προωθούνται στο ερώτημα που εκτελείται στη βάση δεδομένων, με αποτέλεσμα να φορτώνονται λιγότερα δεδομένα στο Spark.

3.2 Apache Sedona

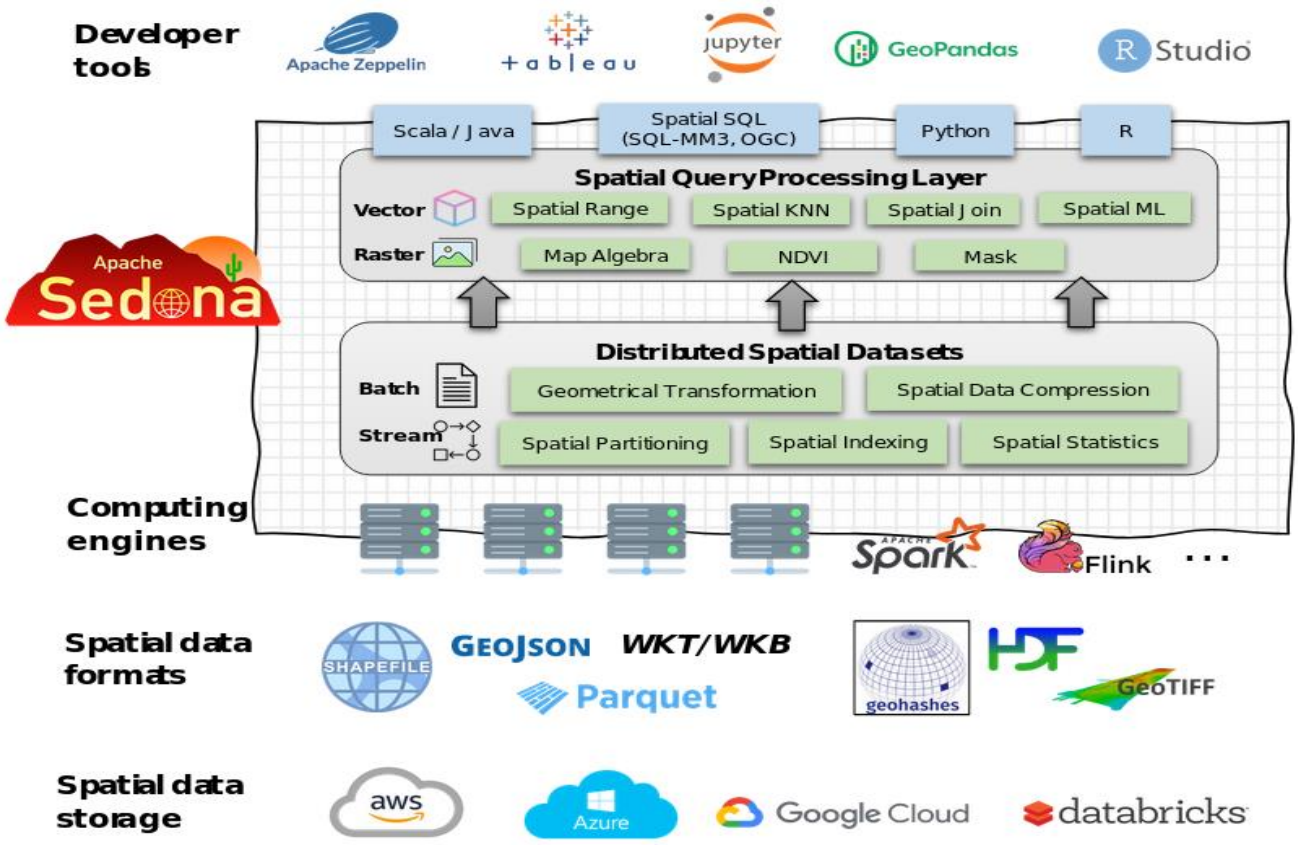
Το Apache Sedona (πρώην GeoSpark) είναι ένα ανοικτού κώδικα κατανεμημένο σύστημα για επεξεργασία χωρικών δεδομένων μεγάλης κλίμακας. Είναι χτισμένο πάνω στο Apache Spark και επεκτείνει τις δυνατότητές του για την υποστήριξη τύπων και λειτουργιών χωρικών δεδομένων. Το Apache Sedona παρέχει μια ευρεία γκάμα λειτουργιών και αλγορίθμων χωρικής ανάλυσης για την επεξεργασία και ανάλυση μεγάλων όγκων χωρικών δεδομένων. Το Sedona έχει σχεδιαστεί για να είναι κλιμακωτό, αποδοτικό και εύκολο στη χρήση [19, 20].

3.2.1 Αρχιτεκτονική

Το Apache Sedona ακολουθεί μια παρόμοια αρχιτεκτονική με το Apache Spark. Το σύστημα αποτελείται από έναν driver που διαχειρίζεται την εκτέλεση εργασιών σε ένα cluster. Ο driver καθορίζει τις λειτουργίες που πρέπει να εκτελεστούν στα δεδομένα και στη συνέχεια αποστέλλει τις εργασίες στους executors για εκτέλεση. Κάθε executor διαθέτει μια μονάδα επεξεργασίας και μνήμης για την εκτέλεση των εργασιών.

Η βασική δομή δεδομένων στο Apache Sedona είναι τα Spatial Resilient Distributed Datasets (SpatialRDDs), τα οποία είναι επέκταση των RDDs του Spark και επιτρέπουν την αναπαράσταση διαφορετικών τύπων χωρικών

δεδομένων, όπως σημεία, γραμμές, πολύγωνα και κύκλους. Τα SpatialRDDs χρησιμοποιούνται για την εισαγωγή, ανάκτηση, και επεξεργασία χωρικών δεδομένων. Επιπλέον, έχουν υλοποιημένες μεθόδους και τελεστές που υποστηρίζουν την εκτέλεση ερωτημάτων σχετικά με την χωρική τοποθεσία όπως, k-nearest neighbors (knn) queries, range queries, spatial joins και άλλα.



Εικόνα 8: Αρχιτεκτονική Apache Sedona (source)

3.2.2 Spatial Partitioning - Indexing

Η επεξεργασία χωρικών δεδομένων αφορά τη διαχείριση μεγάλων συνόλων δεδομένων καθώς και την αντιμετώπιση πολύπλοκων χωρικών ερωτημάτων. Το SpatialRDD του Apache Sedona αποτελεί μια δομή δεδομένων που μπορεί να εφαρμόζει τεχνικές indexing και partitioning με σκοπό τη βελτιστοποίηση της απόδοσης των ερωτημάτων. Το partitioning περιλαμβάνει την κατάτμηση των δεδομένων σε μικρότερα τμήματα, τα οποία μπορούν να επεξεργαστούν

παράλληλα σε διαφορετικούς executors. Το Apache Sedona υποστηρίζει δύο διαφορετικές τεχνικές partitioning, το KDB-tree και το Quad-tree:

- Το KDB-tree (K-Dimensional Binary Tree) partitioning είναι μια μέθοδος που χρησιμοποιείται για την κατάτμηση ενός χώρου υψηλής διαστατικότητας σε μικρότερα τμήματα. Αυτό επιτυγχάνεται αναδρομικά, διχοτομώντας τον χώρο σε δύο μέρη μέσω ενός άξονα και μιας τιμής διαχωρισμού, μέχρι κάθε τμήμα να περιέχει έναν μικρό αριθμό από αντικείμενα.
- Το Quad-tree χωρίζει έναν δισδιάστατο χώρο σε μικρότερα τμήματα, μέχρι κάθε τμήμα να περιέχει έναν περιορισμένο αριθμό αντικειμένων. Αυτό επιτυγχάνεται αναδρομικά, κατατμίζοντας τον χώρο σε ίσα τεταρτημόρια.

Το Apache Sedona επίσης υποστηρίζει διάφορες στρατηγικές indexing, οι οποίες μπορούν να βελτιώσουν σημαντικά την επεξεργασία των δεδομένων και την εκτέλεση των ερωτημάτων. Η επιλογή του ευρετηρίου εξαρτάται από το σύνολο των δεδομένων και την περίπτωση χρήσης. Τα ευρετήρια που υποστηρίζονται είναι τα R-tree, KD-tree και KDB-tree:

- Ο R-tree index οργανώνει τα δεδομένα σε ένα ιεραρχικό δέντρο, διαμερίζοντας τον χώρο σε επιμέρους τμήματα.
- Ο KD-tree, k-dimensional tree index, οργανώνει τα δεδομένα σε ένα δυαδικό δέντρο όπου ο κάθε κόμβος αντιστοιχεί σε ένα σημείο κ-διαστάσεων.
- Ο KDB-tree, k-dimensional box tree index, αποτελεί μια παραλλαγή του kd-tree, οργανώνοντας τα δεδομένα σε υπερ-επίπεδα αντί για σημεία.

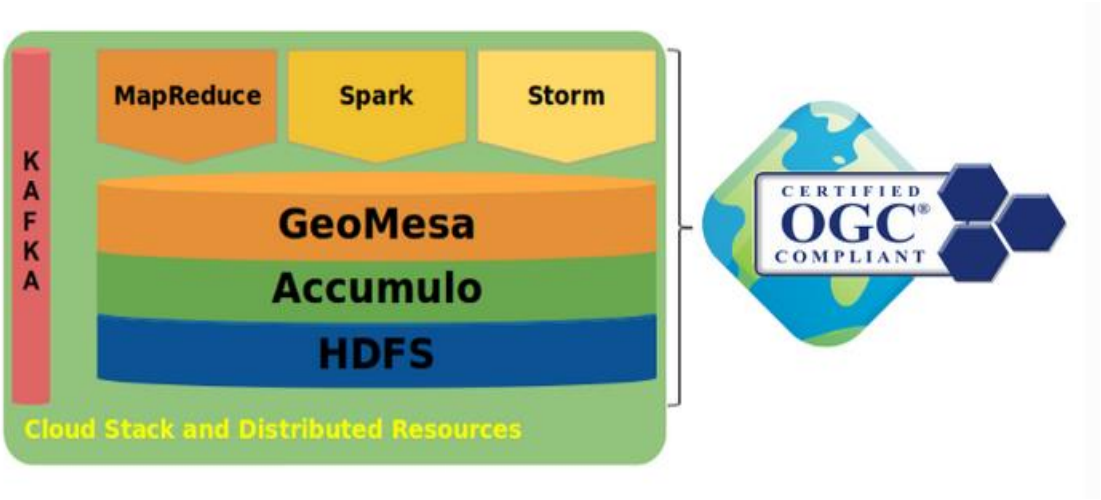
Το Apache Sedona υποστηρίζει indexes σε local και global επίπεδο. Ο global index χρησιμοποιείται σε όλο το σύνολο των δεδομένων, ενώ ο local index χρησιμοποιείται σε επίπεδο executor. Αυτά τα δύο ευρετήρια μπορεί να είναι διαφορετικού τύπου, ώστε να εξυπηρετούνται διάφορες περιπτώσεις χρήσης.

3.3 GeoMesa

Το GeoMesa αποτελεί μια επέκταση ανοιχτού κώδικα του Apache Spark, η οποία δίνει τη δυνατότητα εκτέλεσης ερωτημάτων χωρικής και χωρο-χρονικής ανάλυσης μεγάλης κλίμακας σε κατανεμημένα υπολογιστικά συστήματα. Παρέχει λειτουργίες χωρο-χρονικής ευρετηρίασης στις βάσεις δεδομένων HBase, Accumulo και Cassandra, επιτρέποντας τη μαζική αποθήκευση χωρικών δεδομένων όπως σημεία, γραμμές και πολύγωνα. Επιπλέον, προσφέρει τη δυνατότητα επεξεργασίας ροών χωρο-χρονικών δεδομένων μέσω διασύνδεσης με Apache Kafka. Τέλος, προσφέρει τη δυνατότητα διασύνδεσης με υπάρχοντα συστήματα χαρτογράφησης μέσω των διεπαφών του Open Geospatial Consortium (OGC) και πρωτοκόλλων όπως τα WFS και WMS [22].

3.3.1 Αρχιτεκτονική

Το GeoMesa είναι χτισμένο σε διαφορετικά επίπεδα, τα οποία λειτουργούν συνδυαστικά. Στο κατώτερο επίπεδο, βρίσκεται μια NoSQL column store κατανεμημένη βάση δεδομένων, όπως το HBase, Accumulo ή Cassandra, η οποία επιτρέπει την αποθήκευση χωρικών και χωρο-χρονικών δεδομένων. Ακολουθεί το επίπεδο ευρετηρίασης, το οποίο παρέχει διαφορετικές τεχνικές ευρετηρίασης για χωρικά και χωρο-χρονικά δεδομένα. Επίσης, δίνει τη δυνατότητα δημιουργίας custom indexes σε χαρακτηριστικά πέρα των χωρο-χρονικών. Το τρίτο επίπεδο αφορά την εκτέλεση ερωτημάτων, όπου το GeoMesa επεκτείνει τη λειτουργικότητα του Spark με τη βιβλιοθήκη GeoTools, επιτρέποντας την εκτέλεση χωρικών ερωτημάτων όπως spatial joins, range queries, distance queries κλπ. Το τελικό επίπεδο λειτουργικότητας του GeoMesa παρέχει διασυνδεσιμότητα με μια πληθώρα από άλλα συστήματα μέσω OGC APIs, με Geoserver καθώς και με Apache Kafka για επεξεργασία ροών δεδομένων.



Εικόνα 9: Αρχιτεκτονική GeoMesa (source)

3.3.2 Spatial Partitioning – Indexing

Το GeoMesa υποστηρίζει δύο διαφορετικές τεχνικές partitioning, το quad-tree, όπως και το Apache Sedona, και την καμπύλη Hilbert.

- Η καμπύλη Hilbert μετασχηματίζει έναν πολυδιάστατο χώρο σε μια μονοδιάστατη καμπύλη, με τα κοντινά σημεία στον αρχικό χώρο να αντιστοιχίζονται σε κοντινά σημεία στην καμπύλη. Αυτό επιτυγχάνεται διαιρώντας αναδρομικά τον χώρο σε ίσα τεταρτημόρια, όπου το κάθε τεταρτημόριο μπορεί να αντιστοιχηθεί με μια μοναδική θέση στην καμπύλη. Η αντιστοίχιση κάθε κελιού στην καμπύλη καθορίζεται από την σειρά περιήγησης των τεταρτημόριων. Η σειρά αυτή ορίζεται από την καμπύλη και διασφαλίζει ότι τα γειτονικά κελιά αντιστοιχίζονται σε κοντινές θέσεις στην καμπύλη.

Το GeoMesa υποστηρίζει εξίσου αποδοτικά τα χωρικά και τα χωρο-χρονικά δεδομένα. Για αυτόν τον λόγο, διαθέτει διαφορετικές τεχνικές ευρετηρίασης για κάθε περίπτωση: για τα χωρικά δεδομένα υποστηρίζει τους indexes Z2 και XZ2, ενώ για τα χωρο-χρονικά χρησιμοποιεί τους indexes Z3 και XZ3.

- οι Z2 και Z3 indexes χρησιμοποιούν την καμπύλη Hilbert ως τεχνική partitioning. Οι indexes χρησιμοποιούν μια δισδιάστατη και τρισδιάστατη καμπύλη Z-order αντίστοιχα για να ευρετηριάσουν τα χωρικά και χωρο-χρονικά πεδία.

- οι XZ2 και XZ3 βασίζονται στην μέθοδο partitioning quad-tree. Για την ευρετηρίαση των πεδίων χρησιμοποιούν την τεχνική XZ-ordering, η οποία είναι μια επέκταση της καμπύλης Z-order και έχει σχεδιαστεί για μη σημειακές γεωμετρίες όπως polygons.

4. Περιγραφή Υλοποίησης

Σε αυτό το κεφάλαιο περιγράφεται η υλοποίηση του κατανεμημένου spatio-textual index.

4.1 Κατανεμημένος Spatio-Texual Index

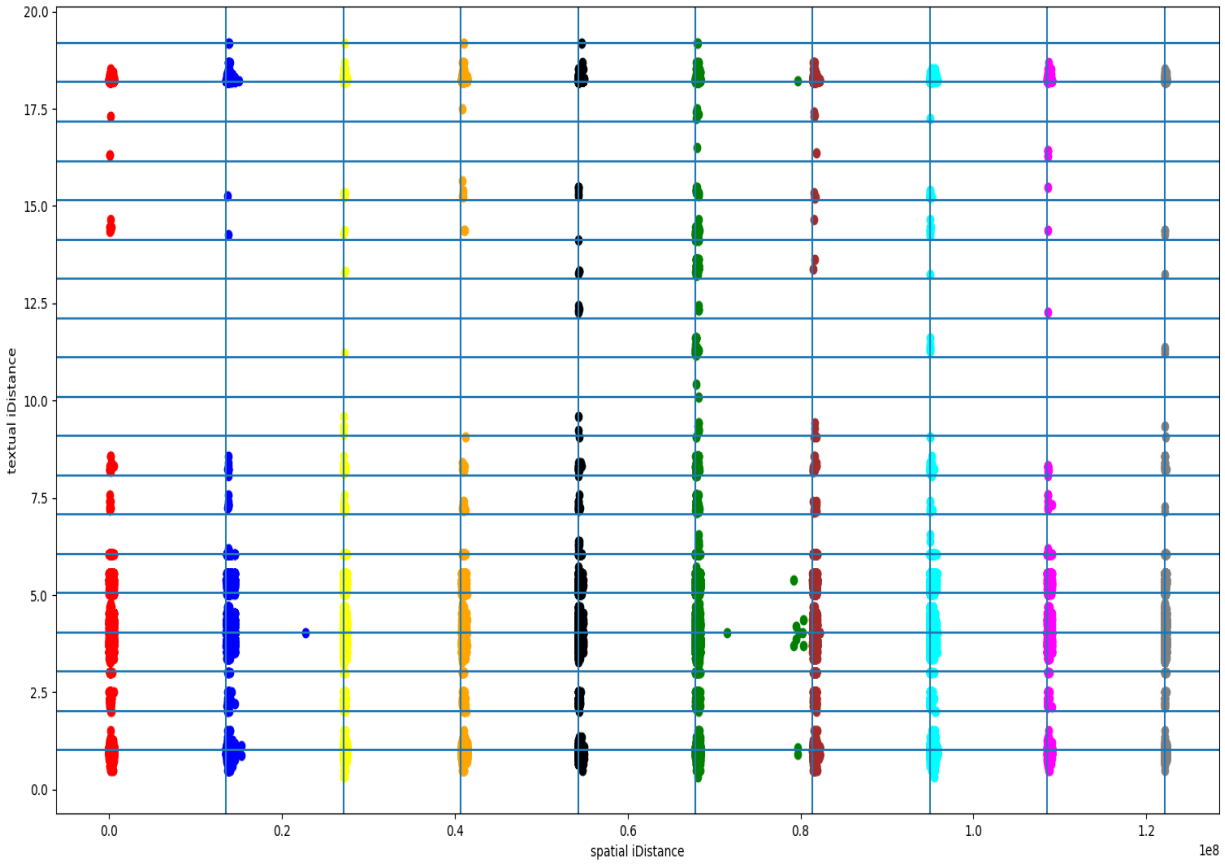
Ο σκοπός του index είναι να μετασχηματίσει ένα πολυδιάστατο σύνολο δεδομένων, σε ένα σύνολο δεδομένων με δύο διαστάσεις και να οργανώσει τα δεδομένα με τέτοιο τρόπο, ώστε να είναι αποδοτικά κατά την αναζήτηση αντικειμένων σε αυτά. Είναι σημαντικό τα δεδομένα να οργανωθούν με τρόπο που να επιτυγχάνει την ισοκατανομή τους στις μονάδες επεξεργασίας, καθώς ο index είναι κατανεμημένος. Η ισοκατανομή των δεδομένων διευκολύνει την οριζόντια κλιμάκωση, δηλαδή όταν αυξάνεται ο αριθμός των μονάδων επεξεργασίας, ο χρόνος εκτέλεσης μειώνεται γραμμικά. Με αυτόν τον τρόπο, η διαδικασία αναζήτησης αντικειμένων θα είναι αποδοτική και κλιμακώσιμη στο πλαίσιο της κατανεμημένης αρχιτεκτονικής.

Αυτό επιτυγχάνεται σε δύο στάδια: το πρώτο αφορά την προεπεξεργασία, κατά την οποία τα δεδομένα οργανώνονται στο δίσκο και δημιουργείται ένας πίνακας με μεταδεδομένα στη μνήμη, όπου αποθηκεύονται πληροφορίες σχετικά με τα χωρικά και κειμενικά clusters του αρχικού χώρου καθώς και σε ποιά partitions αυτά μετασχηματίζονται στον μετασχηματισμένο χώρο. Αυτό το βήμα πρέπει να εκτελεστεί μία φορά πριν από την εκτέλεση οποιουδήποτε ερωτήματος.

Το δεύτερο βήμα αφορά την εκτέλεση ενός ερωτήματος εύρους. Δεδομένου ενός ερωτήματος q που αποτελείται από μια τοποθεσία (q.x, q.y) και ενός συνόλου λέξεων-κλειδιών Q, ο σκοπός είναι να ανακτηθούν όλα τα χωρο-κειμενικά αντικείμενα που βρίσκονται σε απόσταση r από το q και έχουν ομοιότητα στο σύνολο των λέξεων-κλειδιών τους μεγαλύτερη από την τιμή t. Χρησιμοποιώντας τον πίνακα μεταδεδομένων, ένα ερώτημα μπορεί να αναγνωρίσει ποια αρχεία χρειάζεται να φορτωθούν στη μνήμη και να αναζητήσει σε αυτά τα αντικείμενα που πληρούν την παραπάνω συνθήκη.

4.1.1 Pre-Processing Step

Κατά το στάδιο της προεπεξεργασίας, για κάθε αντικείμενο ενός συνόλου δεδομένων υπολογίζονται οι τιμές iDistance για τη χωρική και την κειμενική πληροφορία, όπως περιγράφονται στην εργασία [2] και αναλύονται στο κεφάλαιο 2.2. Η χωρική πληροφορία μετασχηματίζεται στον άξονα x, ενώ η κειμενική μετασχηματίζεται στον άξονα y. Στον μετασχηματισμένο χώρο, τα αντικείμενα οργανώνονται σε ορθογώνια partitions.



Εικόνα 10: Ορθογώνια partitions στον μετασχηματισμένο χώρο

Στην παραπάνω εικόνα παρουσιάζεται ο μετασχηματισμένος χώρος ενός συνόλου δεδομένων, ο οποίος περιλαμβάνει δέκα χωρικά partitions στον άξονα x και είκοσι κειμενικά partitions στον άξονα y.

Για την αποτελεσματική οργάνωση των δεδομένων, τα χωρικά partitions του μετασχηματισμένου χώρου οργανώνονται σε Hive partitions σε ένα

κατανεμημένο σύστημα αρχείων. Ένα Hive partition αντιπροσωπεύει έναν κατάλογο στο σύστημα αρχείων, όπου αποθηκεύονται τα δεδομένα που έχουν την ίδια τιμή σε ένα χαρακτηριστικό τους. Στην συγκεκριμένη περίπτωση, το χαρακτηριστικό είναι το χωρικό cluster id του αρχικού χώρου, και όλα τα δεδομένα που ανήκουν στον ίδιο χωρικό cluster τοποθετούνται στον ίδιο κατάλογο. Τα δεδομένα εντός κάθε Hive partition ταξινομούνται βάση του κειμενικού partition στο οποίο ανήκουν, δηλαδή του κειμενικού cluster id του αρχικού χώρου. Αυτή η ιεραρχική οργάνωση των δεδομένων μέσα σε κάθε Hive partition επιτρέπει την δημιουργία μιας δομής δύο επιπέδων όπου στο πρώτο επίπεδο αναπαριστάται το χωρικό cluster και στο δεύτερο το κειμενικό cluster. Η εκτέλεση κατανεμημένων ερωτημάτων χρησιμοποιώντας μια τέτοια δομή είναι εξίσου αποδοτική τόσο για ερωτήματα που αφορούν την αναζήτηση σε πολλαπλά χωρικά partitions όσο και για ερωτήματα που αφορούν την αναζήτηση σε πολλαπλά κειμενικά partitions.

Για την αποθήκευση των δεδομένων επιλέχθηκε το Parquet format. Το Parquet είναι ένας ανοιχτού κώδικα column-oriented τύπος αποθήκευσης αρχείων που έχει σχεδιαστεί για να παρέχει υψηλή απόδοση και αποτελεσματική αποθήκευση σε δεδομένα μεγάλου όγκου.

Ένα από τα κύρια πλεονεκτήματα του Parquet είναι η αποθήκευση των δεδομένων ανά στήλες. Αυτή η δομή επιτρέπει στο Parquet να επιτύχει υψηλή συμπίεση των δεδομένων, καθώς οι στήλες περιέχουν τον ίδιο τύπο δεδομένων. Μειώνοντας το μέγεθος των αρχείων, μπορεί να εξοικονομηθεί χώρος αποθήκευσης και να βελτιωθεί η απόδοση των ερωτημάτων πάνω στα δεδομένα.

Επιπλέον, το Parquet διατηρεί μεταδεδομένα για κάθε αρχείο, τα οποία περιλαμβάνουν στατιστικά για κάθε στήλη, όπως η ελάχιστη και μέγιστη τιμή της. Αυτό το χαρακτηριστικό, σε συνδυασμό με τη δυνατότητα του Apache Spark να εφαρμόζει predicate push-down φίλτρα σε αρχεία Parquet, χρησιμοποιείται από τον index για να εκτελεί data-skipping. Αυτό σημαίνει ότι φορτώνονται στη μνήμη μόνο τα αρχεία που χρειάζονται, δηλαδή τα δεδομένα που αντιστοιχούν στα χωρικά και κειμενικά clusters, που είναι τα υποψήφια αποτελέσματα, βάσει των κειμενικών και χωρικών κριτηρίων του ερωτήματος.

Τα hive partitions στον δίσκο εμφανίζονται με την παρακάτω μορφή:

```
/spatioTextualIndex/data/cluster_id=0
/spatioTextualIndex/data/cluster_id=1
/spatioTextualIndex/data/cluster_id=2
/spatioTextualIndex/data/cluster_id=3
/spatioTextualIndex/data/cluster_id=4
/spatioTextualIndex/data/cluster_id=5
/spatioTextualIndex/data/cluster_id=6
/spatioTextualIndex/data/cluster_id=7
/spatioTextualIndex/data/cluster_id=8
/spatioTextualIndex/data/cluster_id=9
```

Μέσα σε κάθε hive partition βρίσκονται τα Parquet αρχεία που περιέχουν τα δεδομένα.

```
/spatioTextualIndex/data/cluster_id=0/part-00000-5210acd6-d5d0-4d5c-
be75-c3bfd1771406.c000.snappy.parquet
/spatioTextualIndex/data/cluster_id=0/part-00001-5210acd6-d5d0-4d5c-
be75-c3bfd1771406.c000.snappy.parquet
/spatioTextualIndex/data/cluster_id=0/part-00002-5210acd6-d5d0-4d5c-
be75-c3bfd1771406.c000.snappy.parquet
/spatioTextualIndex/data/cluster_id=0/part-00003-5210acd6-d5d0-4d5c-
be75-c3bfd1771406.c000.snappy.parquet
/spatioTextualIndex/data/cluster_id=0/part-00004-5210acd6-d5d0-4d5c-
be75-c3bfd1771406.c000.snappy.parquet
/spatioTextualIndex/data/cluster_id=0/part-00005-5210acd6-d5d0-4d5c-
be75-c3bfd1771406.c000.snappy.parquet
...
```

Τα Parquet αρχεία σε κάθε hive partition είναι ταξινομημένα βάση της τιμής του κειμενικού partition που ανήκουν, δηλ. στο part-00000 αρχείο θα βρίσκονται οι εγγραφές που περιέχουν τη μικρότερη τιμή, στο part-00001 οι εγγραφές με την αμέσως μεγαλύτερη κοκ.

Επιλέον, ο αλγόριθμος στο βήμα του pre-processing δημιουργεί έναν πίνακα μεταδεδομένων που περιέχει σημαντικές πληροφορίες για τα χωρικά και κειμενικά clusters του αρχικού χώρου καθώς και σε ποιά partitions αντιστοιχούν στον μετασχηματισμένο χώρο. Αυτός ο πίνακας αποθηκεύει το centroid κάθε χωρικού cluster στον αρχικό χώρο, την ακτίνα r του, τις λέξεις-κλειδιά που χαρακτηρίζουν κάθε κειμενικό cluster του αρχικού χώρου, καθώς και τα μοναδικά αναγωριστικά των partitions του μετασχηματισμένου χώρου. Ο

πίνακας μεταδεδομένων διατηρείται στη μνήμη για γρήγορη πρόσβαση και χρησιμοποιείται από κάθε ερώτημα ώστε να εντοπίζει ποία αρχεία χρειάζεται να διαβάσει.

centroid.x	centroid.y	r	keywords	spatial_id	textual_id
0.883	0.499	0.22	creole| greek| dominican	0	0
0.883	0.499	0.22	pets_dogs | italian	0	1
0.116	0.499	0.23	creole| greek| dominican	1	0

Πίνακας 1: Πίνακας μεταδεδομένων spatio-textual index

4.1.2 Query Step

Κατά την εκτέλεση ενός ερωτήματος μέσω του πίνακα μεταδεδομένων που βρίσκεται στη μνήμη, ο αλγόριθμος αρχικά εντοπίζει σε ποια partitions απαιτείται να γίνει αναζήτηση. Αυτό επιτυγχάνεται με την εύρεση των χωρικών clusters που τέμνονται με τη χωρική πληροφορία του ερωτήματος, των κειμενικών clusters που περιέχουν τις λέξεις-κλειδιά του ερωτήματος και σε ποιά partitions αντιστοιχούν αυτά στον μετασχηματισμένο χώρο. Τα μοναδικά αναγνωριστικά των partitions χρησιμοποιούνται ως φίλτρο κατά τη διάρκεια της ανάγνωσης των αρχείων από τον δίσκο. Το Spark στην συνέχεια εκτελεί predicate push-down τα φίλτρα και φορτώνει στη μνήμη μόνο εκείνα τα αρχεία που είναι απαραίτητα για την εκτέλεση του ερωτήματος.

```
spark.read
   .parquet(filesPath)
   .filter(col("spatial_id").isin(spatial_ids))
   .filter(col("textual_id").isin(textual_ids))
```

Η εκτέλεση παραπάνω εντολής ανάγνωσης των Parquet αρχείων από τον δίσκο δημιουργεί το παρακάτω πλάνο εκτέλεσης.

```
(1) Scan parquet
Output [6]: [textual_id#156L, spatial_id#157, lon#158, lat#159,
keywords#165]
Batched: false
Location: InMemoryFileIndex [hdfs://node1:9000/spatioTextualIndex/data]
PartitionFilters: [spatial_id#170 IN (0)]
PushedFilters: [IsNotNull(lat), IsNotNull(lon), In(textual_id, [0,1)]
ReadSchema:
struct<textual_id:bigint,spatial_id:bigint,lon:string,lat:string,keyword
s:array<string>>
```

Το πλάνο εκτέλεσης περιλαμβάνει την ενέργεια ανάγνωσης από αρχεία Parquet. Τα αναγνωριστικά των χωρικών partitions ενσωματώνονται ως PartitionFilters, ενώ τα αναγνωριστικά των κειμενικών partitions ενσωματώνονται ως PushedFilters, όπως φαίνεται στο query plan. Το Apache Spark θα αναζητήσει μόνο στα directories που περιέχονται στα PartitionFilters και θα διαβάσει τα μεταδεδομένα όλων των αρχείων Parquet που βρίσκονται σε αυτά. Έπειτα, θα φορτώσει στη μνήμη μόνο τα αρχεία Parquet που περιέχουν τις τιμές που αντιστοιχούν στα PushedFilters, δηλαδή τα ids των κειμενικών partitions.

Τέλος, στα αρχεία που θα φορτωθούν στη μνήμη από το Apache Spark, πρέπει να εξεταστεί ποιες εγγραφές περιέχονται σε αυτά που ικανοποιούν τη χωρική και κειμενική συνθήκη του ερωτήματος. Συγκεκριμένα, πρέπει η απόσταση από το q να είναι μικρότερη ή ίση του r και η κειμενική ομοιότητα να είναι μεγαλύτερη ή ίση του t. Για το σκοπό αυτό, χρησιμοποιείται η ευκλείδια απόσταση και ο δείκτης Jaccard similarity, οι οποίοι εφαρμόζονται ως φίλτρα για να εξαιρεθούν τα αντικείμενα που δεν ικανοποιούν τη χωρική και κειμενική συνθήκη του ερωτήματος αντίστοιχα. Όλα τα αντικείμενα που παραμένουν μετά την εκτέλεση των φίλτρων αποτελούν το αποτέλεσμα του ερωτήματος.

1. **Input:** q:query, metadataTable:table with metadata info, r:distance threshold, t:textual similarity threshold
2. **Output**: results: set of spatio-textual objects
3. *spatialPartitions* = **findSpatialPartitions**(q.x,q.y,r,metadataTable)
4. *textualPartitions* = **findTextualPartitions**(q.keywords,metadataTable)
5. *filesToLoad* = **scanParquetFiles**(*spatialPartitions*, *textualPartitions*)
6. *results* = *filesToLoad*.filter(**EuclideanDistance**(r)).filter(**JaccardSimilarity**(t))
7. return *results*

Πίνακας 2: Ψευδοκώδικας - εκτέλεση ερωτήματος

Στον παραπάνω ψευδοκώδικα βλέπουμε ότι ένα ερώτημα δέχεται ως είσοδο μια τοποθεσία, ένα σύνολο λέξεων-κλειδιων, τον πίνακα μεταδεδομένων, την απόσταση r και την κειμενική ομοιότητα t. Αρχικά εντοπίζει τα partitions που θα γίνει η αναζήτηση. Για τα χωρικά partitions καλείται η μέθοδος findSpatialPartitions, η οποία εντοπίζει σε ποιά χωρικά clusters του αρχικού χώρου τέμνεται η χωρική πληροφορία του ερωτήματος και επιστρέφει τα χωρικά partitions που αντιστοιχούν σε αυτά. Για τα κειμενικά partitions χρησιμοποιείται η μέθοδος findTextualPartitions, η οποία εντοπίζει τα κειμενικά clusters του αρχικού χώρου που περιέχουν τις λέξεις-κλειδιά του ερωτήματος και επιστρέφει σε ποιά κειμενικά partitions μετασχηματίζονται. Στην συνέχεια το ερώτημα, περνάει ως παραμέτρους τα ids που έχει βρεί στην μέθοδο scanParquetFiles η οποία φορτώνει στη μνήμη τα αρχεία που περιέχουν δεδομένα από αυτά τα partitions. Τέλος, σε όλα τα δεδομένα που θα φορτωθούν στη μνήμη, θα εξεταστεί η απόσταση από το q και η κειμενική ομοιότητα ως προς το σύνολο των λέξεων-κλειδιων.

5. Πειραματική Διαδικασία και Αποτελέσματα

Σε αυτό το κεφάλαιο παρουσιάζονται τα αποτελέσματα των πειραμάτων. Στην ενότητα 5.1 γίνεται μια σύντομη περιγραφή σχετικά με το πειραματικό setup, ενώ στην ενότητα 5.2 γίνεται η παρουσίαση των αποτελεσμάτων.

5.1 Περιγραφή Πειραμάτων

Τα πειράματα εκτελέστηκαν σε υποδομή cluster με 32GB μνήμη, 20 cores και 80GB δίσκο. Χρησιμοποιήθηκε ένα σύνολο με πραγματικά δεδομένα, το οποίο περιλαμβάνει geo-tagged tweets με πέντε εκατομμύρια εγγραφές, και δημιουργήθηκαν σύνολα από συνθετικά δεδομένα που αποτελούνται από δέκα και είκοσι εκατομμύρια εγγραφές.

Για την παραγωγή των συνθετικών δεδομένων χρησιμοποιήθηκε μια γεννήτρια χωρο-κειμενικών δεδομένων, η οποία δημιουργεί εγγραφές με το παρακάτω σχήμα.

Πεδίο	Περιγραφή
id	Αύξων ακέραιος αριθμός που λειτουργεί ως αναγνωριστικό μια εγγραφής
longitude	Δεκαδικός αριθμός που συμβολίζει το γεωγραφικό μήκος
latitude	Δεκαδικός αριθμός που συμβολίζει το γεωγραφικό πλάτος
keywords	Κείμενο που αποτελείται από 1 έως 200 διαφορετικές λέξεις

Πίνακας 3: Σχήμα συνθετικών δεδομένων

Το πεδίο id αντιπροσωπεύει έναν αύξοντα ακέραιο αριθμό. Τα πεδία longitude και latitude ακολουθούν την κανονική κατανομή με μέση τιμή 0.5 και τυπική απόκλιση 0.25, και οι τιμές τους ανήκουν στο διάστημα [0,1]. Το πεδίο keywords

περιλαμβάνει ένα κείμενο μεταξύ 1 και 200 διαφορετικών λέξεων, χωρισμένες με κόμμα, από το λεξιλόγιο που χρησιμοποιήθηκε. Το πλήθος των λέξεων κάθε εγγραφής επιλέχθηκε τυχαία μέσω μια γεννήτριας τυχαίων αριθμών. Τέλος, το λεξιλόγιο αποτελείται από λέξεις που επιλέγονται τυχαία από ένα προκαθορισμένο σύνολο λέξεων.

Οι μέθοδοι που συγκρίθηκαν είναι :

- Spatial First αλγόριθμος, ο οποίος έχει υλοποιηθεί στο Apache Spark. Αυτός ο αλγόριθμος αρχικά υπολογίζει τα αποτελέσματα με βάση το χωρικό κριτήριο του ερωτήματος, χρησιμοποιώντας την ευκλείδια απόσταση. Στη συνέχεια, για τα αντικείμενα που ικανοποιούν το χωρικό κριτήριο, υπολογίζεται η κειμενική ομοιότητα σε σχέση με τις λέξεις-κλειδιά του ερωτήματος, χρησιμοποιώντας τη μετρική Jaccard similarity. Τα δεδομένα που διαβάζει είναι αποθηκευμένα σε CSV μορφή.
- Textual First αλγόριθμος, ο οποίος έχει υλοποιηθεί στο Apache Spark. Αυτός ο αλγόριθμος αρχικά υπολογίζει τα αποτελέσματα βάσει της κειμενικής πληροφορίας και στη συνέχεια λαμβάνει υπόψη τη χωρική πληροφορία. Για την αξιολόγηση της κειμενικής πληροφορίας χρησιμοποιείται η μετρική Jaccard similarity, ενώ για την χωρική απόσταση χρησιμοποιείται η ευκλείδια απόσταση. Τα δεδομένα που διαβάζει είναι αποθηκευμένα σε CSV μορφή.
- Spatial First αλγόριθμος, υλοποιημένος σε Apache Sedona. Ο αλγόριθμος spatial first υπολογίζει πρώτα τα αποτελέσματα βάσει της χωρικής πληροφορίας. Ωστόσο, καθώς είναι υλοποιημένος σε Apache Sedona, χρησιμοποιεί το προεπιλεγμένο χωρικό ευρετήριο του Sedona, το οποίο είναι το quad-tree, και τα δεδομένα γίνονται partitioning με την τεχνική kdb-tree. Αυτό σημαίνει ότι τα αντικείμενα που ικανοποιούν το χωρικό κριτήριο θα ανακτηθούν πιο γρήγορα σε σχέση με τεχνικές που δεν χρησιμοποιούν χωρικό ευρετήριο, καθώς δεν απαιτείται έλεγχος όλων των δεδομένων, αλλά μόνο αυτών που βρίσκονται στα partitions που επιλέγονται βάση της χωρικής πληροφορίας του ερωτήματος. Στη συνέχεια, όλα τα αποτελέσματα που ικανοποιούν τη χωρική συνθήκη ελέγχονται βάσει της κειμενικής τους ομοιότητας με τις λέξεις-κλειδιά του ερωτήματος, χρησιμοποιώντας το Jaccard similarity.

- Spatial First αλγοριθμος, υλοποιημένος σε GeoMesa. Ο αλγόριθμος Spatial First εκτελεί πρώτα τον υπολογισμό των αποτελεσμάτων βάσει της χωρικής πληροφορίας. Ωστόσο, λόγω της υλοποίησής του στην πλατφόρμα GeoMesa, χρησιμοποιεί το προκαθορισμένο χωρικό ευρετήριο του GeoMesa, που είναι το Z2, και εφαρμόζει την τεχνική Hilbert curve για τη διαμέριση των δεδομένων. Η χρήση αυτών των τεχνικών οδηγεί σε πιο αποδοτική αναζήτηση των αντικειμένων που πληρούν το χωρικό κριτήριο σε σύγκριση με τεχνικές που δεν χρησιμοποιούν χωρικά ευρετήρια, καθώς απαιτείται ο έλεγχος μόνο των κελιών που είναι εντός της ενδιαφερόμενης περιοχής, και όχι όλων των δεδομένων. Στη συνέχεια, ελέγχονται όλα τα αποτελέσματα που πληρούν τη χωρική συνθήκη ως προς την κειμενική τους ομοιότητα προς τις λέξεις-κλειδιά του ερωτήματος, με τη χρήση της μετρικής Jaccard Similarity, για την εξαγωγή των τελικών αποτελεσμάτων.
- Αλγόριθμος που βασίζεται στον κατανεμημένο spatio-textual index, όπως περιγράφηκε στην ενότητα 4, υλοποιημένος σε Apache Spark.

Στο πλαίσιο των πειραμάτων, δοκιμάστηκαν διαφορετικές τιμές για τις παρακάτω παραμέτρους:

- Distance threshold (r): 0.1, 0.5, 0.9
- Textual similarity (t): 0.1, 0.3, 0.5
- Set of keywords (Q): most_frequent_keywords, least_frequent_keywords

Για την απόσταση και την κειμενική ομοιότητα εξετάστηκαν τρεις τιμές: δύο ακραίες τιμές και μια ενδιάμεση. Ως προς τις λέξεις-κλειδιά, έχουν εξαχθεί από κάθε σύνολο δεδομένων οι περισσότερο και οι λιγότερο συχνά εμφανιζόμενες λέξεις.

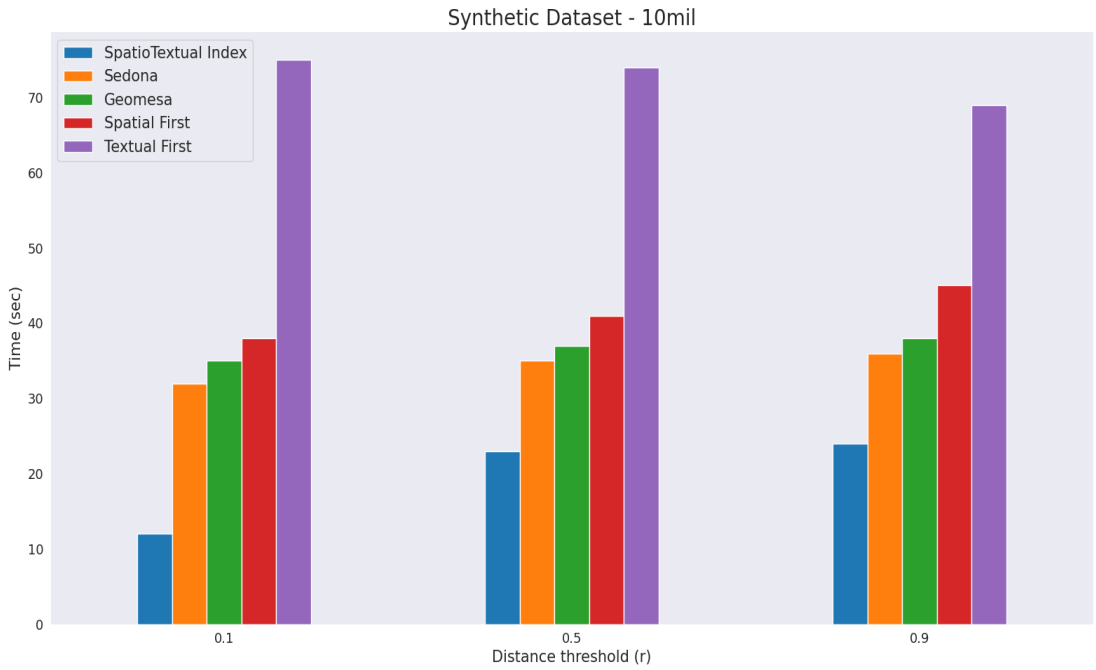
Η μετρική που χρησιμοποιήθηκε για την αξιολόγηση των αποτελεσμάτων ήταν ο χρόνος εκτέλεσης του ερωτήματος. Κάθε ερώτημα εκτελέστηκε 20 φορές και υπολογίστηκε η μέση τιμή του χρόνου εκτέλεσης.

5.2 Αποτελέσματα

Τα βασικά στοιχεία ενός ερωτήματος μεταβάλλονται στα πειράματα για να διαπιστωθεί κατά πόσο επηρεάζεται ο χρόνος εκτέλεσης κάθε αλγορίθμου και να συγκριθούν μεταξύ τους σε όλες τις περιπτώσεις.

5.2.1 Μεταβολή r

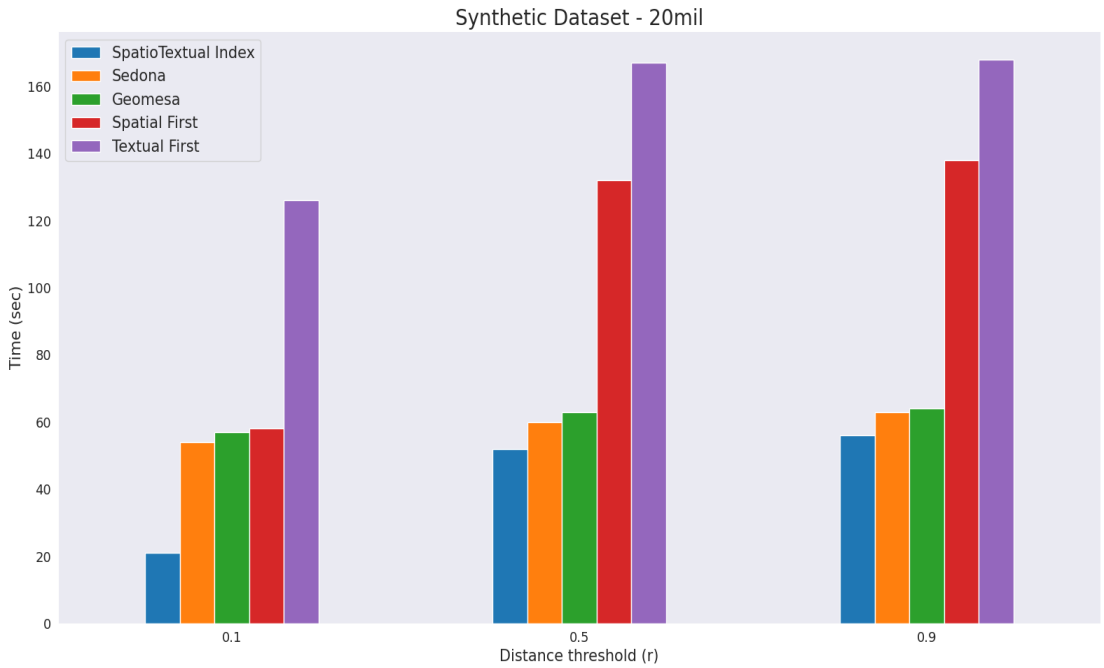
Στο πρώτο πείραμα μεταβάλλεται η τιμή της ακτίνας r, ενώ οι υπόλοιπες παράμετροι παραμένουν σταθερές. Η κειμενική ομοιότητα έχει την τιμή 0.1 και για λέξεις-κλειδιά χρησιμοποιήθηκαν οι πιο συχνά εμφανιζόμενες λέξεις.



Εικόνα 11: Μεταβολή τιμής r - συνθετικά δεδομένα 10εκ

Για τα συνθετικά δεδομένα που περιέχουν δέκα εκατομμύρια εγγραφές, ο spatio-textual index έχει χρόνο εκτέλεσης 12 δευτερόλεπτα για r=0.1, ενώ για r=0.9 αυξήθηκε στα 24 δευτερόλεπτα. Η υλοποίηση spatial first σε Apache Sedona παρουσίασε τον δεύτερο καλύτερο χρόνο εκτέλεσης, με μια αύξηση της διάρκειας από 50% έως 270% σε σχέση με τον index. Ο πιο αργός χρόνος

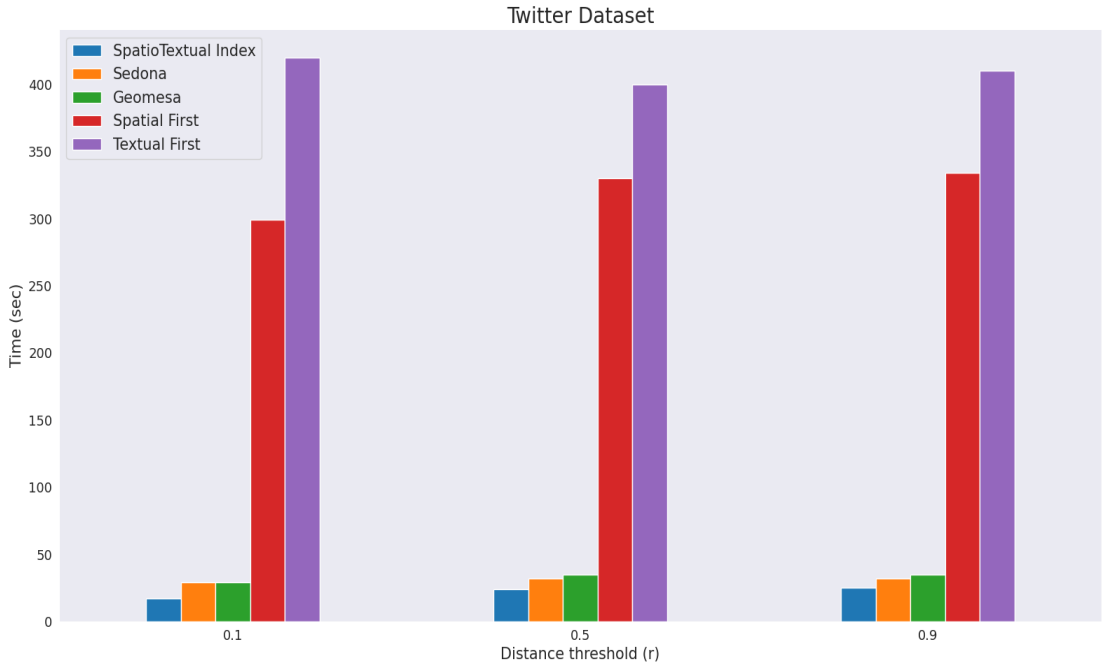
εκτέλεσης αφορά στην textual first υλοποίηση σε Apache Spark με 75 δευτερόλεπτα για r=0.1, 74 δευτερόλεπτα για r=0.5 και 69 δευτερόλεπτα για r=0.9. Τέλος, παρατηρείται ότι με την αύξηση της τιμής του r, πραγματοποιείται αντίστοιχη αύξηση του χρόνου εκτέλεσης σε όλες τις περιπτώσεις. Αυτή η τάση είναι αναμενόμενη, καθώς με την αύξηση του r, το ερώτημα επιστρέφει μεγαλύτερο όγκο δεδομένων προς επεξεργασία.



Εικόνα 12: Μεταβολή τιμής r - συνθετικά δεδομένα 20εκ

Για τα συνθετικά δεδομένα που περιλαμβάνουν είκοσι εκατομμύρια εγγραφές, παρατηρούμε ότι ο spatio-textual index παρουσιάζει χρόνους εκτέλεσης στα 21 δευτερόλεπτα για r=0.1 και αυξάνεται έως τα 56 δευτερόλεπτα για r=0.9. Ο δεύτερος καλύτερος χρόνος εκτέλεσης είναι η spatial first υλοποίηση σε Apache Sedona, η οποία είναι από 15% έως 160% πιο αργή σε σχέση με τον index. Ο πιο αργός χρόνος εκτέλεσης παρατηρείται στην υλοποίηση textual first στο Apache Spark, με τιμές που κυμαίνονται από 126 δευτερόλεπτα για r=0.1 έως 168 δευτερόλεπτα για r=0.9. Κατά τον διπλασιασμό του μεγέθους των δεδομένων στο πείραμα, παρατηρήθηκε σχεδόν παρόμοιος διπλασιασμός των χρόνων εκτέλεσης για τον spatio-textual index. Αυτό υποδεικνύει μια γραμμική

αύξηση του χρόνου εκτέλεσης ως αποτέλεσμα της αυξημένης ποσότητας δεδομένων που πρέπει να επεξεργαστεί ο spatio-textual index.

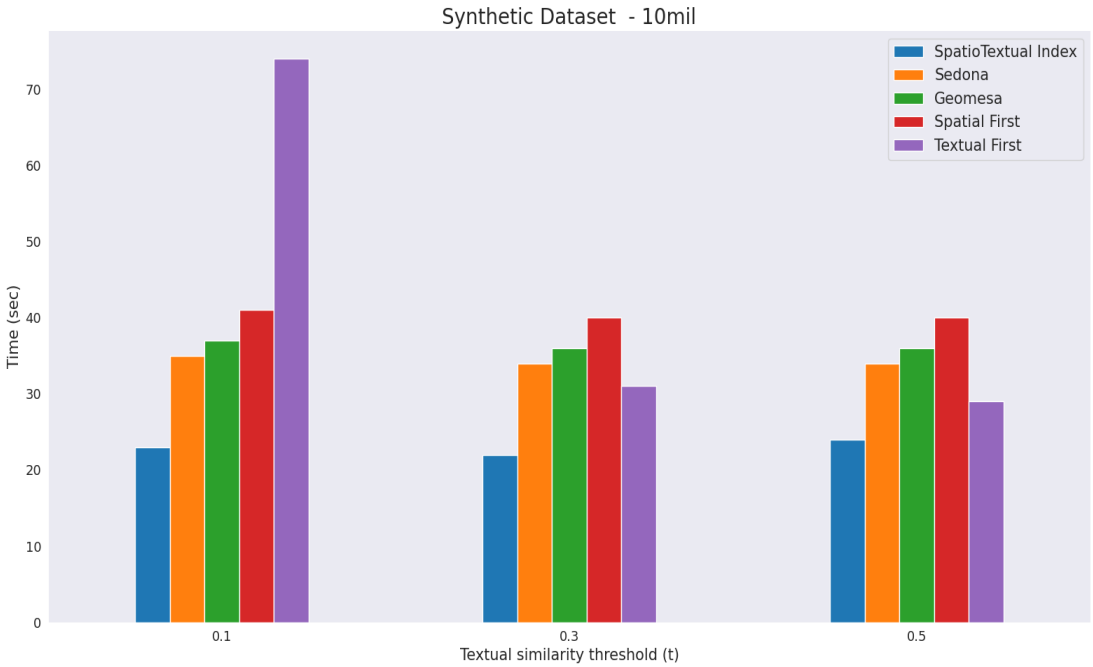


Εικόνα 13: Μεταβολή τιμής r - σύνολο δεδομένων Twitter

Για το σύνολο δεδομένων του Twitter, ο spatio-textual index έχει τους καλύτερους χρόνους εκτέλεσης, με τιμές που κυμαίνονται από 17 έως 25 δευτερόλεπτα. Η υλοποίηση spatial first στο Apache Sedona αποτελεί τον δεύτερο καλύτερο χρόνο εκτέλεσης και είναι από 30% έως 70% πιο αργή σε σχέση με τον index. Οι υλοποιήσεις spatial first και textual first στο Apache Spark είναι σημαντικά πιο αργές από τις άλλες μεθόδους, με χρόνους εκτέλεσης που υπερβαίνουν τα 300 δευτερόλεπτα. Στη διεξαγωγή του πειράματος με χρήση πραγματικών δεδομένων, διαπιστώνεται σημαντική διαφορά στους χρόνους εκτέλεσης μεταξύ των μεθόδων που χρησιμοποιούν κάποια μορφή index και των μεθόδων που δεν χρησιμοποιούν. Αυτό οφείλεται στο ότι τα πραγματικά δεδομένα δεν ακολουθούν κάποια συγκεκριμένη κατανομή με συνέπεια να μην κατανέμονται τα δεδομένα δίκαια στις μονάδες επεξεργασίας. Αυτό το φαινόμενο επηρεάζει δραματικά την απόδοση της κατανεμημένης επεξεργασίας.

5.2.2 Μεταβολή t

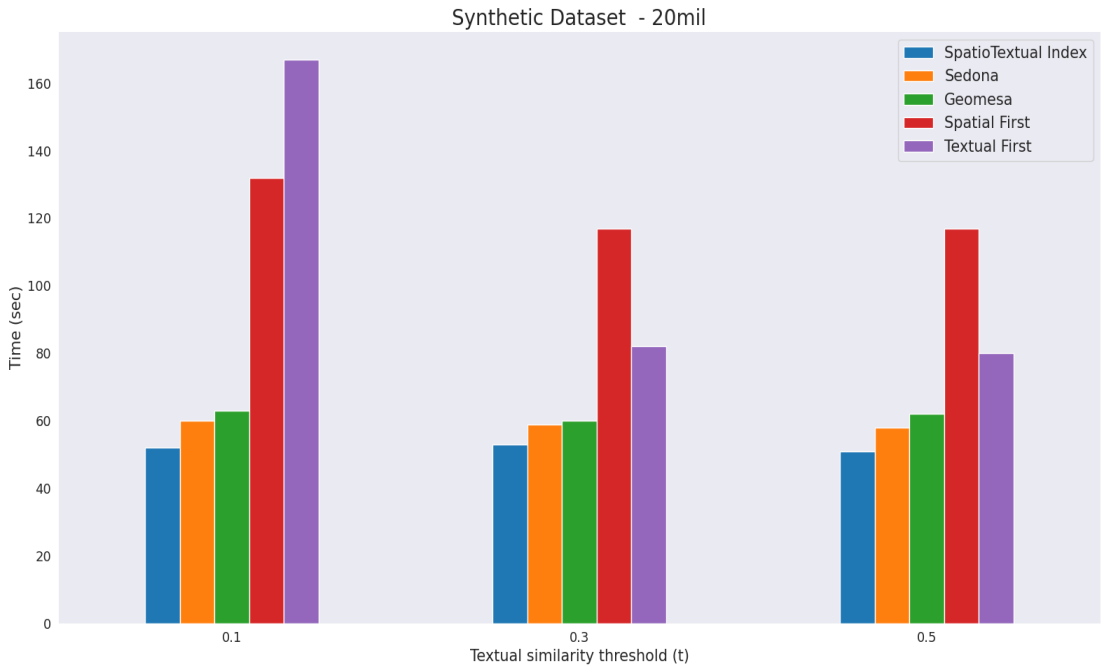
Στο δεύτερο πείραμα, μεταβάλλεται η τιμή της κειμενικής ομοιότητας t, ενώ οι υπόλοιπες παράμετροι παραμένουν σταθερές. Η ακτίνα r έχει την τιμή 0.5 και για λέξεις-κλειδιά χρησιμοποιήθηκαν οι πιο συχνά εμφανιζόμενες λέξεις.



Εικόνα 14: Μεταβολή τιμής t - συνθετικά δεδομένα 10εκ

Για τα συνθετικά δεδομένα που περιέχουν δέκα εκατομμύρια εγγραφές, ο spatio-textual index παρουσιάζει έναν σχεδόν σταθερό χρόνο εκτέλεσης από 22 έως 24 δευτερόλεπτα για κάθε τιμή του t. Ο δεύτερος καλύτερος χρόνος εκτέλεσης είναι η spatial first υλοποίηση σε Apache Sedona, η οποία είναι περίπου 60% πιο αργή σε σχέση με τον index για t=0.1, ενώ για τις τιμές 0.3 και 0.5 είναι η textual first υλοποίηση σε Apache Spark που είναι 50% πιο αργή από τον index. Ο χειρότερος χρόνος εκτέλεσης για t=0.1 αφορά στην textual first υλοποίηση σε Apache Spark με 74 δευτερόλεπτα, ενώ για τις τιμές t=0.3 και t=0.5 αφορά στην υλοποίηση spatial first σε Apache Spark με χρόνους εκτέλεσης στα 40 δευτερόλεπτα. Η αλλαγή της τιμής του t δεν επηρεάζει τον χρόνο εκτέλεσης για

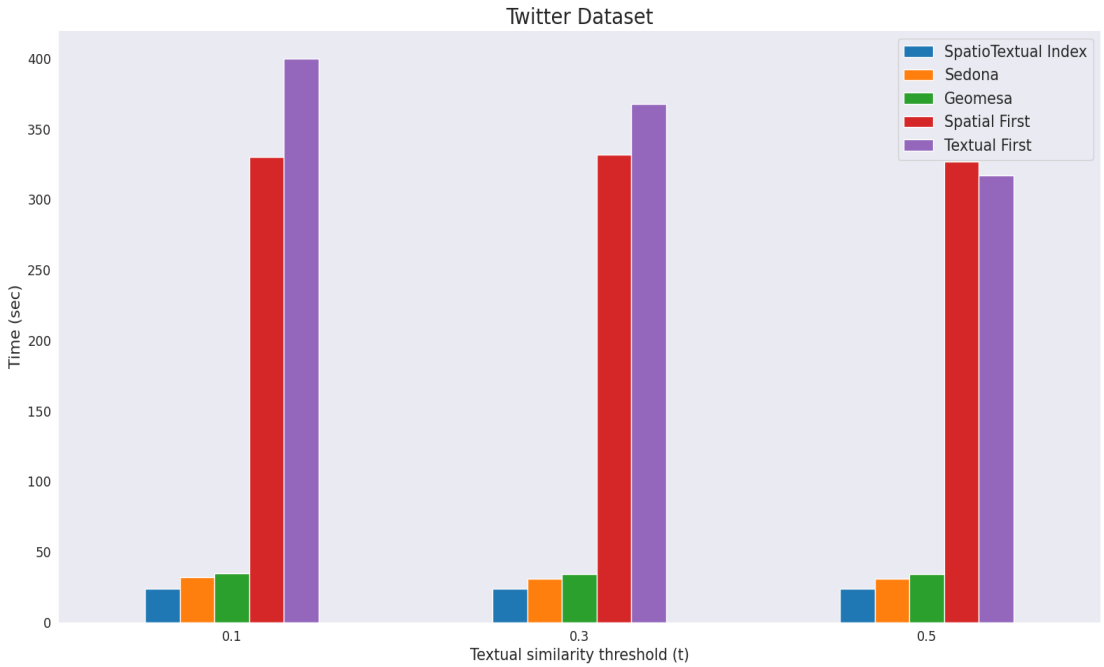
τον spatio-textual index, αντίθετα επηρεάζει σημαντικά τον textual first αλγόριθμο, ο οποίος είναι ο δεύτερος καλύτερος χρόνος εκτέλεσης για τις μεγαλύτερες τιμές του t. Αυτό θεωρείται λογικό καθώς ο textual first χρησιμοποιεί την κειμενική ομοιότητα σαν πρωτο κριτήριο. Με την αύξηση της τιμής του t, ο αριθμός των δεδομένων που υποβάλλονται σε επεξεργασία μειώνεται. Από την άλλη πλευρά, ο spatio-textual index χρησιμοποιεί και τα δύο κριτήρια, αλλά η κειμενική ομοιότητα δεν επηρεάζει τη συμπεριφορά του καθώς δεν μεταβάλει τον αριθμό των αρχείων που θα διαβαστούν από τον δίσκο.



Εικόνα 15: Μεταβολή τιμής t - συνθετικά δεδομένα 20 εκ

Για τα συνθετικά δεδομένα που περιέχουν είκοσι εκατομμύρια εγγραφές, ο spatio-textual index παρουσιάζει έναν σταθερό χρόνο εκτέλεσης από 51 έως 53 δευτερόλεπτα για τις μεταβολές της τιμής του t. Ο δεύτερος καλύτερος χρόνος εκτέλεσης παρατηρείται στην spatial first υλοποίηση σε Apache Sedona, η οποία είναι περίπου 20% πιο αργή σε σχέση με τον index. Ο πιο αργός χρόνος εκτέλεσης για t=0.1 αφορά στην textual first υλοποίηση σε Apache Spark με 167 δευτερόλεπτα, ενώ για τις τιμές t=0.3 και t=0.5 αφορά την υλοποίηση spatial first σε Apache Spark με χρόνους εκτέλεσης στα 117 δευτερόλεπτα. Κατά τον

διπλασιασμό του μεγέθους των δεδομένων στο πείραμα, παρατηρήθηκε σχεδόν αντίστοιχος διπλασιασμός του χρόνου εκτέλεσης για τον ο spatio-textual index. Αυτό υποδηλώνει την γραμμικότητα του χρόνου εκτέλεσης ως προς τον αριθμό των δεδομένων, και σε αυτό το πείραμα.

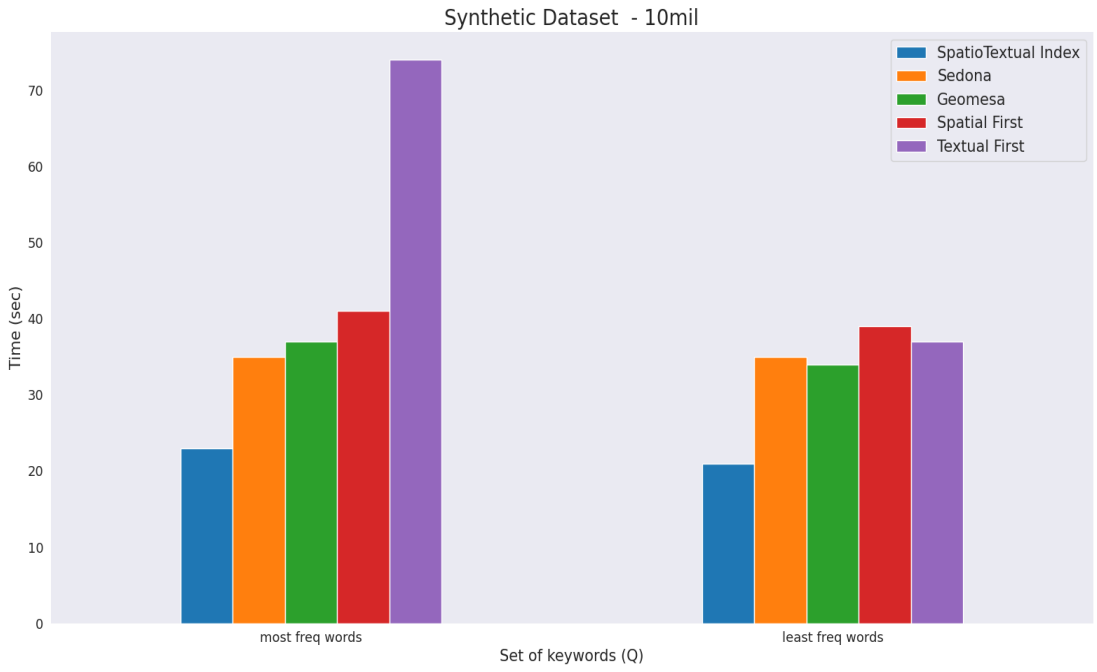


Εικόνα 16: Μεταβολή τιμής t - σύνολο δεδομένων Twitter

Για το σύνολο δεδομένων του Twitter, παρατηρείται ότι ο spatio-textual index διατηρεί σταθερό χρόνο εκτέλεσης στα 24 δευτερόλεπτα για κάθε τιμή του t. Η spatial first υλοποίηση σε Apache Sedona είναι 33% πιο αργή σε σχέση με τον index. Η textual first υλοποίηση σε Apache Spark παρουσιάζει τον πιο αργό χρόνο εκτέλεσης για t=0.1 και t=0.3, ενώ για t=0.5 η πιο αργή εκτέλεση παρατηρείται στην υλοποίηση spatial first σε Apache Spark. Και στο παρόν πείραμα, επιβεβαιώνεται η διαφορά στην αποδοση μεταξύ των μεθόδων που υιοθετούν κάποια μορφή ευρετηρίασης και των μεθόδων που δεν την εφαρμόζουν.

5.2.3 Μεταβολή keywords

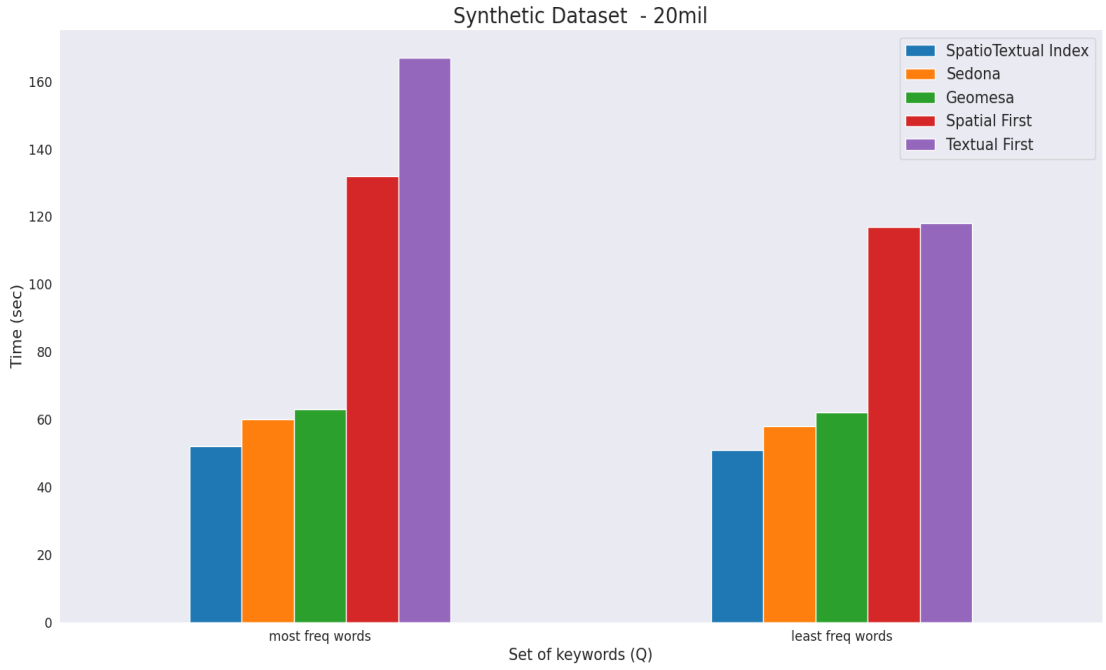
Στο τρίτο πείραμα μεταβάλλονται οι λέξεις-κλειδιά. Χρησιμοποιήθηκε ένα σύνολο με τις πιο συχνά εμφανιζόμενες λέξεις-κλειδιά και ένα με τις λιγότερο εμφανιζόμενες. Η ακτίνα r έχει την τιμή 0.5 και η κειμενική ομοιότητα έχει την τιμή 0.1. Το σύνολο που περιλαμβάνει τις λιγότερο συχνά εμφανιζόμενες λέξεις-κλειδιά αναμένεται ότι θα διαθέτει βελτιωμένους χρόνους εκτέλεσης σε σχέση με το σύνολο των συχνότερα εμφανιζόμενων λέξεων-κλειδιών, καθώς μετά την εφαρμογή του φίλτρου κειμενικής πληροφορίας, ο όγκος των δεδομένων προς επεξεργασία θα είναι μικρότερος. Αυτό προκύπτει λόγω ότι τα αντικείμενα που συσχετίζονται με τις λιγότερο συχνές λέξεις-κλειδιά είναι λιγότερα στο σύνολο των δεδομένων.



Εικόνα 17: Μεταβολή των λέξεων-κλειδιών - συνθετικά δεδομένα 10 εκ

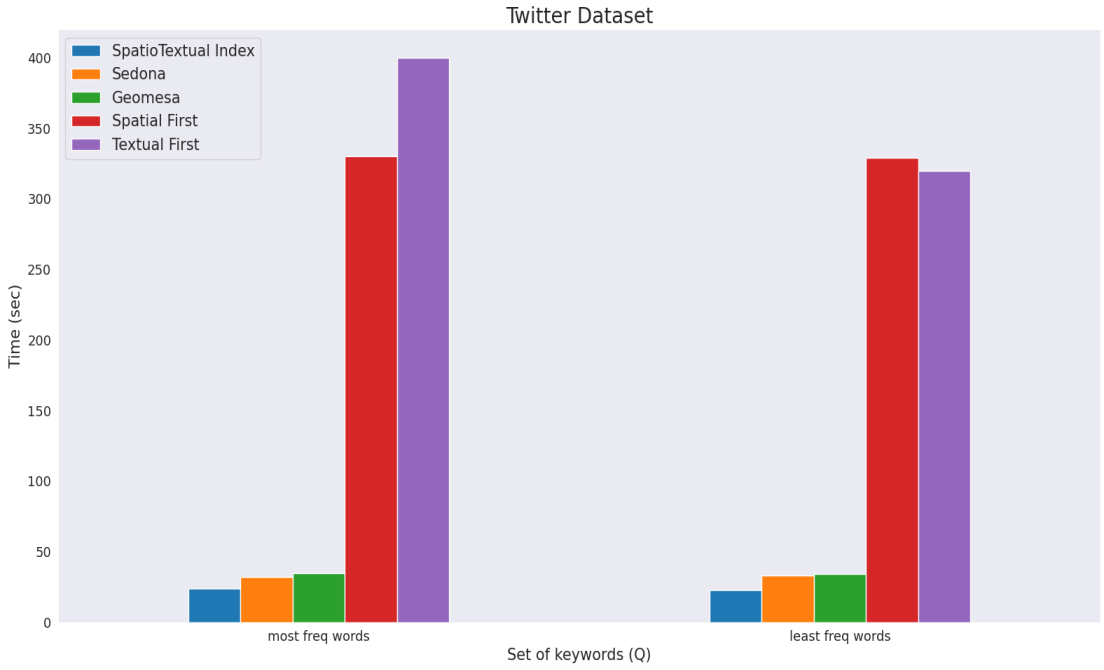
Για τα συνθετικά δεδομένα με δέκα εκατομμύρια εγγραφές, ο spatio-textual index παρουσιάζει τον καλύτερο χρόνο εκτέλεσης, ο οποίος κυμαίνεται από 21 έως 23 δευτερόλεπτα. Η δεύτερη καλύτερη επίδοση για το σύνολο με τις συχνότερα εμφανιζόμενες λέξεις-κλειδιά ανήκει στην υλοποίηση spatial first στο Apache

Sedona, με 35 δευτερόλεπτα, ενώ για το σύνολο με τις λιγότερο συχνά εμφανιζόμενες λέξεις-κλειδιά ο δεύτερος καλύτερος χρόνος εκτέλεσης ανήκει στην υλοποίηση spatial first στο GeoMesa, με 34 δευτερόλεπτα. Ο χειρότερος χρόνος εκτέλεσης για το σύνολο με τις συχνότερα εμφανιζόμενες λέξεις-κλειδιά ανήκει στην υλοποίηση textual first στο Apache Spark, με 74 δευτερόλεπτα. Για το σύνολο με τις λιγότερο συχνά εμφανιζόμενες λέξεις-κλειδιά ο πιο αργός χρόνος εκτέλεσης ανήκει στην υλοποίηση spatial first στο Apache Spark, με 39 δευτερόλεπτα. Είναι αναμενόμενο ότι ο αλγόριθμος textual first θα παρουσιάσει υψηλότερο χρόνο εκτέλεσης στο σύνολο δεδομένων με τις πιο συχνά εμφανιζόμενες λέξεις, καθώς η προτεραιότητα δίνεται στο κειμενικό φίλτρο και απομένουν πολλές εγγραφές για επεξεργασία με βάση το χωρικό φίλτρο. Ωστόσο, ο αλγόριθμος textual first παρουσιάζει σημαντική βελτίωση όταν το σύνολο δεδομένων μεταβαίνει στις λιγότερο συχνά εμφανιζόμενες λέξεις-κλειδιά, καθώς τα αντικείμενα που απομένουν μετά την εφαρμογή του κειμενικού φίλτρου είναι σημαντικά λιγότερα.



Εικόνα 18: Μεταβολή των λέξεων- κλειδιών - συνθετικά δεδομένα 20 εκ

Όσον αφορά τα συνθετικά δεδομένα με είκοσι εκατομμύρια εγγραφές, ο spatio-textual index παρουσιάζει τον καλύτερο χρόνο εκτέλεσης, ο οποίος είναι σχεδόν σταθερός με 51 δευτερόλεπτα για το σύνολο με τις λιγότερο εμφανιζόμενες λέξεις-κλειδιά και 52 δευτερόλεπτα για το σύνολο με τις πιο συχνά εμφανιζόμενες λέξεις-κλειδιά. Ο δεύτερος καλύτερος χρόνος εκτέλεσης αφορά την υλοποίηση spatial first στο Apache Sedona, με χρόνους εκτέλεσης από 58 έως 60 δευτερόλεπτα. Οι πιο αργοί χρόνοι εκτέλεσης αφορούν στην υλοποίηση textual first στο Apache Spark με χρόνους εκτέλεσης από 117 έως 167 δευτερόλεπτα. Παρατηρείται ότι η αλλαγή στο σύνολο των λέξεων-κλειδιών δεν έχει σημαντικό αντίκτυπο στο χρόνο εκτέλεσης καμίας μεθόδου, με εξαίρεση την υλοποίηση textual-first. Σχετικά με τον index, αυτό αποδίδεται στην παρατήρηση ότι αν και το σύνολο των λέξεων κλειδιών μεταβάλλεται, τα αρχεία που προορίζονται για ανάγνωση από τον δίσκο δεν επηρεάζονται σημαντικά. Αυτό οφείλεται στο ότι το μέγεθος του συνόλου δεδομένων είναι σχετικά μικρό και στο στάδιο της προεπεξεργασίας έχουν δημιουργηθεί μόνο λίγα Parquet αρχεία για κάθε hive partition.



Εικόνα 19: Μεταβολή των λέξεων- κλειδιών - σύνολο δεδομένων Twitter

Για το σύνολο δεδομένων του Twitter, ο spatio-textual index επιδεικνύει τους καλύτερους χρόνους εκτέλεσης, οι οποίοι κυμαίνονται από 23 έως 24 δευτερόλεπτα. Η υλοποίηση spatial first στο Apache Sedona έχει τον δεύτερο καλύτερο χρόνο εκτέλεσης και είναι περίπου 40% πιο αργή από τον index. Οι υλοποιήσεις spatial first και textual first στο Apache Spark παρουσιάζουν πολύ μεγάλους χρόνους εκτέλεσης, που υπερβαίνουν τα 300 δευτερόλεπτα. Παρατηρείται και σε αυτό το πείραμα η μεγάλη διαφορά σε απόδοση που υπάρχει στις μεθόδους spatial first και textual first του Apache Spark όταν το σύνολο αφορά πραγματικά δεδομένα, σε σχέση με τις υπόλοιπες που χρησιμοποιούν κάποια μορφή index.

6. Συμπεράσματα - Μελλοντικές Επεκτάσεις

Σε αυτό το κεφάλαιο ολοκληρώνεται η διπλωματική εργασία με μια σύντομη αναφορά στα αποτελέσματα και τη συμβολή της, καθώς και με προτάσεις σχετικά με μελλοντικές επεκτάσεις.

6.1 Συμπεράσματα

Στο πλαίσιο της παρούσας εργασίας, παρουσιάστηκε ένας αλγόριθμος ευρετηρίασης χωρο-κειμενικών δεδομένων για εκτέλεση ερωτημάτων εύρους σε κατανεμημένο περιβάλλον επεξεργασίας. Ο αλγόριθμος υλοποιήθηκε σε Apache Spark και συγκρίθηκε με μεθόδους spatial first και textual first, υλοποιημένες στο Apache Spark, Apache Sedona και GeoMesa ως προς τον χρόνο εκτέλεσης.

Ο αλγόριθμος επεκτείνει τη μέθοδο που προτείνεται στην εργασία [2] για τη μετατροπή ενός πολυδιάστατου χώρου σε δισδιάστατο, αποθηκεύει το σύνολο δεδομένων σε μορφή Parquet και εκμεταλλεύεται τη δυνατότητα του Apache Spark να κάνει push down φίλτρα κατά την ανάγνωση αρχείων Parquet. Ο αλγόριθμος διατηρεί έναν πίνακα μεταδεδομένων στη μνήμη ο οποίος περιλαμβάνει πληροφορίες σχετικά με τα clusters του αρχικού χώρου και τον τρόπο αντιστοίχισής τους στον μετασχηματισμένο χώρο. Τέλος, ο αλγόριθμος οργανώνει τα αρχεία Parquet με τέτοιο τρόπο ώστε να εξασφαλίζεται η αποδοτική κατανεμημένη αναζήτηση εγγραφών σε αυτά.

Η πειραματική διαδικασία εκτελέστηκε σε τρία διαφορετικά σύνολα δεδομένων: ένα με πραγματικά δεδομένα από το Twitter, με 5 εκατομμύρια εγγραφές, και δύο με συνθετικά δεδομένα που ακολουθούν την κανονική κατανομή, με 10 και 20 εκατομμύρια εγγραφές αντίστοιχα. Τα αποτελέσματα έδειξαν ότι ο index ήταν σημαντικά ταχύτερος σε σχέση με τις μεθόδους ανταγωνιστές. Συγκεκριμένα, στα συνθετικά δεδομένα με 10 εκατομμύρια εγγραφές, ο index ήταν από 50% έως 270% πιο γρήγορος από τη δεύτερη καλύτερη μέθοδο, ενώ στα σύνολο με τις 20 εκατομμύρια εγγραφές ήταν από 15% έως 160% γρηγορότερος. Όσον

αφορά τα πραγματικά δεδομένα, ο index ήταν από 33% έως 60% πιο γρήγορος σε σχέση με την δεύτερη καλύτερη μέθοδο. Παρατηρούμε ότι όσον αφορά τα πειράματα στα συνθετικά σύνολα δεδομένων, ο χρόνος εκτέλεσης αυξήθηκε σχεδόν γραμμικά σε όλες τις περιπτώσεις, συγκεκριμένα όταν διπλασιάστηκαν οι εγγραφές ο χρόνος εκτέλεσης διπλασιάστηκε σε όλες τις περιπτώσεις.

Τέλος, βάσει των πειραμάτων που εκτελέστηκαν,φαίνεται ότι η παράμετρος της απόστασης r επηρεάζει σημαντικά τον χρόνο εκτέλεσης του index. Από την άλλη πλευρά, οι λέξεις-κλειδιά φαίνεται να τον επηρεάζουν σε μικρότερο βαθμό, ενώ η παράμετρος της κειμενικής ομοιότητας δεν φαίνεται να έχει καμία απολύτως επίπτωση στο χρόνο εκτέλεσης του index.

6.2 Μελλοντικές Επεκτάσεις

Ως μελλοντικές επεκτάσεις προτείνεται η σύγκριση του index με μεθόδους ανταγωνιστές και σε άλλα χωρο-κειμενικά ερωτήματα όπως top knn, boolean knn και box range queries. Επίσης, προτείνεται να υλοποιηθεί και η μετατροπή του πολυδιάστατου χώρου σε δισδιάστατο, όπως περιγράφεται στην εργασία [2], σε κατανεμημένο περιβάλλον επεξεργασίας, όπως το Apache Spark, με σκοπό την ενσωμάτωση του index ως ανεξάρτητης βιβλιοθήκης σε αυτό.

Βιβλιογραφία

[1] Zhida Chen, Lisi Chen, Gao Cong, and Christian S. Jensen. 2021. Location- and keyword-based querying of geo-textual data: a survey. The VLDB Journal 30, 4 (Jul 2021), 603–640. https://doi.org/10.1007/s00778-021-00661-w

[2] Panagiotis Tampakis, Dimitris Spyrellis, Christos Doulkeridis, Nikos Pelekis, Christos Kalyvas, and Akrivi Vlachou. 2021. A Novel Indexing Method for Spatial-Keyword Range Queries. In 17th International Symposium on Spatial and Temporal Databases (SSTD '21). Association for Computing Machinery, New York, NY, USA, 54–63. https://doi.org/10.1145/3469830.3470897

[3] H. V. Jagadish, Beng Chin Ooi, Kian-Lee Tan, Cui Yu, and Rui Zhang. 2005. IDistance: An adaptive B+-tree based indexing method for nearest neighbor search. ACM Trans. Database Syst. 30, 2 (June 2005), 364–397. https://doi.org/10.1145/1071610.1071612

[4] Ariel Cary, Ouri Wolfson, and Naphtali Rishe. 2010. Efficient and scalable method for processing top-k spatial Boolean queries. In Proceedings of the 22nd international conference on Scientific and statistical database management (SSDBM'10). Springer-Verlag, Berlin, Heidelberg, 87–95.

[5] Yen-Yu Chen, Torsten Suel, and Alexander Markowetz. 2006. Efficient query processing in geographic web search engines. In Proceedings of the 2006 ACM SIGMOD international conference on Management of data (SIGMOD '06). Association for Computing Machinery, New York, NY, USA, 277–288. https://doi.org/10.1145/1142473.1142505

[6] Maria Christoforaki, Jinru He, Constantinos Dimopoulos, Alexander Markowetz, and Torsten Suel. 2011. Text vs. space: efficient geo-search query processing. In Proceedings of the 20th ACM international conference on Information and knowledge management (CIKM '11). Association for Computing Machinery, New York, NY, USA, 423–432. https://doi.org/10.1145/2063576.2063641

[7] Gao Cong, Christian S. Jensen, and Dingming Wu. 2009. Efficient retrieval of the top-k most relevant spatial web objects. Proc. VLDB Endow.2,1(August2009),337–348. https://doi.org/10.14778/1687627.1687666

[8] Ian De Felipe, Vagelis Hristidis, and Naphtali Rishe. 2008. Keyword Search on Spatial Databases. In Proceedings of the 2008 IEEE 24th International Conference on Data Engineering (ICDE '08). IEEE Computer Society, USA, 656–665. https://doi.org/10.1109/ICDE.2008.4497474

[9] Zhisheng Li, Ken C. K. Lee, Baihua Zheng, Wang-Chien Lee, Dik Lee, and Xufa Wang. 2011. IR-Tree: An Efficient Index for Geographic Document Search. IEEE Trans. on Knowl. and Data Eng. 23, 4 (April 2011), 585–599. https://doi.org/10.1109/TKDE.2010.149

[10] João B. Rocha-Junior, Orestis Gkorgkas, Simon Jonassen, and Kjetil Nørvåg. 2011. Efficient processing of top-k spatial keyword queries. In Proceedings of the 12th international conference on Advances in spatial and temporal databases (SSTD'11). Springer-Verlag, Berlin, Heidelberg, 205–222.

[11] Ali Khodaei, Cyrus Shahabi, and Chen Li. 2010. Hybrid indexing and seamless ranking of spatial and textual features of web documents. In Proceedings of the 21st international conference on Database and expert systems applications: Part I (DEXA'10). Springer-Verlag, Berlin, Heidelberg, 450–466.

[12] Vaid, S., Jones, C.B., Joho, H., Sanderson, M. (2005). Spatio-textual Indexing for Geographical Search on the Web. In: Bauzer Medeiros, C., Egenhofer, M.J., Bertino, E. (eds) Advances in Spatial and Temporal Databases. SSTD 2005. Lecture Notes in Computer Science, vol 3633. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11535331_13

[13] Dingming Wu, Man Lung Yiu, Gao Cong, and Christian S. Jensen. 2012. Joint Top-K Spatial Keyword Query Processing. IEEE Trans. on Knowl. and Data Eng. 24, 10 (October 2012), 1889–1903. https://doi.org/10.1109/TKDE.2011.172

[14] Yinghua Zhou, Xing Xie, Chuang Wang, Yuchang Gong, and Wei-Ying Ma. 2005. Hybrid index structures for location-based web search. In Proceedings of the 14th ACM international conference on Information and knowledge management (CIKM '05). Association for Computing Machinery, New York, NY, USA, 155–162. https://doi.org/10.1145/1099554.1099584

[15] Apache Spark documentation https://spark.apache.org

[16] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2012. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (NSDI'12). USENIX Association, USA, 2.

[17] Michael Armbrust, Reynold S. Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K. Bradley, Xiangrui Meng, Tomer Kaftan, Michael J. Franklin, Ali Ghodsi, and Matei Zaharia. 2015. Spark SQL: Relational Data Processing in Spark. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15). Association for

Computing Machinery, New York, NY, USA, 1383–1394. https://doi.org/10.1145/2723372.2742797

[18] Apache Sedona documentation https://sedona.apache.org

[19] Jia Yu, Jinxuan Wu, and Mohamed Sarwat. 2015. GeoSpark: a cluster computing framework for processing large-scale spatial data. In Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '15). Association for Computing Machinery, New York, NY, USA, Article 70, 1–4. https://doi.org/10.1145/2820783.2820860

[20] Jia Yu, Zongsi Zhang, and Mohamed Sarwat. 2019. Spatial data management in apache spark: the GeoSpark perspective and beyond. Geoinformatica 23, 1 (January 2019), 37–78. https://doi.org/10.1007/s10707-018-0330-9

[21] GeoMesa documentation https://www.geomesa.org

[22] Hughes, James & Annex, Andrew & Eichelberger, Chris & Fox, Anthony & Hulbert, Andrew & Ronquest, Michael. (2015). GeoMesa: a distributed architecture for spatio-temporal fusion. 94730F. 10.1117/12.2177233.

[23] Matei Zaharia, Bill Champers (2018) Spark: The Definitive Guide, O'Reilly Media,Inc.

[24] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual Web search engine. In Proceedings of the seventh international conference on World Wide Web 7 (WWW7). Elsevier Science Publishers B. V., NLD, 107–117.

[25] Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. 2010. Pregel: a system for large-scale graph processing. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data (SIGMOD '10). Association for Computing Machinery, New York, NY, USA, 135–146. https://doi.org/10.1145/1807167.1807184