ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
Π.Μ.Σ. «ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΥΠΗΡΕΣΙΕΣ»
ΚΑΤΕΥΘΥΝΣΗ: ΔΙΚΤΥΟΚΕΝΤΡΙΚΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

Ανάλυση Στατιστικά Σημαντικών Εστιών για Μεγάλα Δεδομένα

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Παρασκευόπουλος Άρης Ιάκωβος

Επιβλέπων Καθηγητής: Δουλκερίδης Χρήστος

Αθήνα, Φεβρουάριος 2018

University of Piraeus

Department of Digital Systems
Network-Oriented Information Systems

Hot Spot Analysis on Big Data

MASTER GRADUATE THESIS

Paraskevopoulos Aris Iakovos

Supervising Professor: Doulkeridis Christos

Athens, February 2018

Περίληψη

Καθημερινά παράγεται ένας μεγάλος όγκος από χώρο-χρονικά δεδομένα και δεδομένα τροχιών. Αυτά περιγράφουν φαινόμενα και μοτίβα τα οποία μπορούν να αναγνωριστούν μέσω τεχνικών ανάλυσης. Μία από αυτές είναι η Hot Spot ανάλυση όπου έχει στόχο στην εύρεση στατιστικά σημαντικών συγκεντρώσεων από τέτοια φαινόμενα. Κύριος χώρος εφαρμογής της συγκεκριμένης ανάλυσης είναι τα χωρικά δεδομένα.

Σκοπός αυτής τη διπλωματικής εργασίας είναι η περιγραφή του προβλήματος της Hot Spot ανάλυσης πάνω σε χώρο-χρονικά δεδομένα και δεδομένα τροχιών μεγάλης κλίμακας και η πρόταση λύσεων στα παραπάνω προβλήματα μέσω αλγορίθμων παράλληλης επεξεργασίας μεγάλων δεδομένων.

Abstract

Nowdays, large volumes of spatio-temporal data and trajectories are generated. These data describe phenomena and patterns that may be identified through analysis techniques. One of these is the Hot Spot analysis where it aims to find statistically significant clusters of such phenomena. The main area of application of this analysis is spatial data.

The purpose of this diploma thesis is to describe the problem of Hot Spot analysis on large-scale spatio-temporal data and trajectory data and to propose solutions to the above problems through parallel processing algorithms for big data.

Αναλυτικά Περιεχόμενα

1. Εισαγωγή

1.1 Το Πρόβλημα Hot Spot Ανάλυση

Μεγάλος όγκος χώρο-χρονικών δεδομένων και δεδομένων τροχιών δημιουργείται καθημερινά από τροχιές αντικειμένων, μηνύματα και check ins στα μέσα κοινωνικής δικτύωσης και άλλα. Η ανάλυση τέτοιων τύπων δεδομένων λέγεται ότι έχει την δυνατότητα να αποκαλύψει κρυφά μοτίβα και άλλες πληροφορίες οι οποίες πολλές φορές δεν είναι εμφανείς. Έτσι και το Hotspot Analysis είναι μία τεχνική ανάλυσης για την αποτύπωση τέτοιων στατιστικά αξιοσημείωτων φαινομένων συγκεντρωμένα γύρω από μία περιοχή. Πρωτοεμφανίστηκε ως μία μορφή ανάλυσης χωρικών και γεωγραφικών δεδομένων ωστόσο μετέπειτα μεταφέρθηκε και στον τομέα τον χώρο-χρονικών δεδομένων.

Αυτή η μορφή ανάλυσης είναι πολύ συχνή για δεδομένα μεσαίας κλίμακας και χωρικές βάσεις δεδομένων. Όμως ακόμα δεν έχουν γίνει αρκετά βήματα για την βελτιστοποίηση των εργαλείων που υπάρχουν για δεδομένα μεγάλης κλίμακας και χώρο-χρονικά δεδομένα ή τροχιές.

1.2 Στόχος της εργασίας

Αναγνωρίζοντας αυτό το κενό που υπάρχει σε τέτοιου είδους αναλύσεις γίνεται μία προσπάθεια εδώ να δοθεί μία λύση η οποία θα επεξεργάζεται χώρο-χρονικά δεδομένα και τροχιές μεγάλης κλίμακας με αποδοτικό τρόπο, δίνοντας ταυτόχρονα αποτελέσματα τα οποία έχουν κάποια αξία.

Για να λυθεί το παραπάνω πρόβλημα χρησιμοποιήθηκε το Apache Spark μία από τις πιο γρήγορες πλατφόρμες ανάλυσης δεδομένων ανοιχτού κώδικα. Με αυτόν το τρόπο επιτυγχάνεται παράλληλη επεξεργασία δεδομένων, βέλτιστα και με ασφάλεια ως προς τα σφάλματα και τις αστοχίες του υλικού. Οι αλγόριθμοι που υλοποιήθηκαν είναι από τους πρώτους που επεξεργάζονται παράλληλα χώρο-χρονικά δεδομένα και τροχιές με αποδοτικό τρόπο. Η εργασία θα επικεντρωθεί στην παρουσίαση αλγορίθμων που αναπτύχθηκαν για την αντιμετώπιση των παραπάνω προβλημάτων και στην αποδοτικότητα τους.

1.3 Δομή της Εργασίας

Η δομή αυτής της εργασίας περιγράφεται στις παρακάτω γραμμές. Στο κεφάλαιο 2 γίνεται μία αναφορά στα μεγάλα δεδομένα, τα χαρακτηριστικά τους, στους τρόπου διαχείρισης αλλά και στις κατηγορίες που αυτά υπάγονται. Το τρίτο κεφάλαιο είναι αφιερωμένο στα εργαλεία που έχουν δημιουργηθεί για την παράλληλη επεξεργασία μεγάλου όγκου δεδομένων. Εκεί θα αναφερθεί το Apache Hadoop και το Apache Spark, δύο πλατφόρμες ευρέως γνωστές που δίνουν λύση σε προβλήματα παράλληλης επεξεργασίας μεγάλων δεδομένων.

Το κεφάλαιο 4 παρουσιάζει την έννοια του Hotspot Analysis το οποίο είναι και η μεθοδολογία που ακολουθείται από τους αλγορίθμους οι οποίοι παρουσιάζονται στο κεφάλαιο 5. Έπειτα στο κεφάλαιο 6 παρουσιάζονται τα πειραματικά αποτελέσματα της ανάλυσης και γίνεται λόγος για μελλοντικές προεκτάσεις που μπορούν να γίνουν. Ακολουθεί ο επίλογος ο οποίος κλείνει και την εργασία,

2. Μεγάλα Δεδομένα

2.1 Εισαγωγή

Την σημερινή περίοδο τα δεδομένα που παράγονται αυξάνονται με ραγδαίους ρυθμούς. Αυτά τα δεδομένα προέρχονται από διαφόρων ειδών μέσα όπως μέσα κοινωνικής δικτύωσης, διάφορες ηλεκτρονικές συσκευές, αναζητήσεις σε μηχανές αναζήτησης. Εκ των πραγμάτων οποιαδήποτε συσκευή έχει πρόσβαση στο διαδίκτυο μπορεί να παράγει δεδομένα τα οποία θα χρησιμοποιηθούν με κάποιον τρόπο. Λόγοι για την αξιοποίηση αυτών υπάρχουν αρκετοί. Συχνά εταιρίες αναζητούν τρόπους για να αναγνωρίσουν τάσεις και να αυξήσουν την εμπειρία του πελάτη τους και να πάρουν αποφάσεις. Η ανάλυση μεγάλου όγκου δεδομένων μπορεί να επιφέρει θετικά αποτελέσματα ως προς αυτές τις αναζητήσεις. Επίσης αρκετοί είναι οι οργανισμοί οι οποίοι χρησιμοποιούν τα παραγόμενα δεδομένα προς όφελος της κοινωνίας.

Ο λόγος γίνεται για τον όρο Big Data ο οποίος αναφέρεται σε έναν τεράστιο όγκο δεδομένων. Λέγεται ότι πρωτοεμφανίστηκε το 1998 από έναν μηχανικό ηλεκτρονικών υπολογιστών, τον John R. Mashey, σε μία ομιλία [1]. Προφανώς ο όρος τότε είχε χρησιμοποιηθεί για ελαφρώς διαφορετικό σκοπό, ωστόσο προϊδέαζε το κοινό για την ραγδαία αύξηση του όγκου των δεδομένων που θα υπάρξει στο μέλλον. Μετά από αυτήν την αναφορά ο όρος Big Data αργεί να ξαναεμφανιστεί. Στην ακαδημαϊκή και στην εμπορική βιβλιογραφία καταγράφεται μία έκρηξη μετά το 2010, με έναν καταιγισμό αφοσιωμένων άρθρων και αναφορών. Από τότε μέχρι σήμερα προσελκύει τεράστιο ενδιαφέρον και έχει μελετηθεί από ακαδημαϊκούς και εταιρίες.

Από τον όρο Big Data δεν θα μπορούσαν να λείπουν τα χαρακτηριστικά. Πολλές φορές αναφέρονται στα 3 V. Volume, Velocity, Variety. Στο [2] ο Martin Hilbert περιγράφει αναλυτικά τα 3 V. Όπως αναφέρει ο συγγραφέας τα δεδομένα θα παράγονται ότι και αν συμβεί σε μεγάλη κλίμακα και ο ρυθμός παραγωγής τους θα αυξάνεται. Αυτός ο μεγάλος όγκος δεδομένων (Volume) θα αντικαταστήσει τις δειγματοληπτικές μεθόδους της στατιστικής για την μελέτη ενός φαινομένου. Επίσης γράφει στο άρθρο του ότι αυτά τα δεδομένα μπορεί αρκετές φορές να είναι πραγματικού χρόνου (Velocity) όπως τα δεδομένα που παράγονται κάθε δευτερόλεπτο στα μέσα κοινωνικής δικτύωσης. Τέλος τα δεδομένα μπορούν να έχουν πολλές μορφές ή να έχουν συλλεχθεί από διαφορετικές πηγές (Variety).

Καθένα από τα παραπάνω χαρακτηριστικά δημιουργεί και μία πρόκληση κατά την ανάλυση των δεδομένων. Όταν ο όγκος των δεδομένων είναι πολύ μεγάλος μπορεί να μην φτάνει ο αποθηκευτικός χώρος, η RAM ή η υπολογιστική ισχύς ενός cluster. Όταν τα δεδομένα είναι real time μπορεί η ταχύτητα τους να ξεπερνάει την ταχύτητα με την οποία ένας αλγόριθμος τα επεξεργάζεται ούτως ώστε να γίνεται απαραίτητη μία

μέθοδος δειγματοληψίας. Τέλος όταν τα δεδομένα είναι ετερογενή πολλές φορές δεν μπορούν να χρησιμοποιηθούν ταυτόχρονα στην ίδια ανάλυση χωρίς διαδικασίες προ-επεξεργασίας των δεδομένων.

Παρά τις όποιες δυσκολίες στην ανάλυση των μεγάλων δεδομένων έχουν γίνει βήματα ως προς την αποτελεσματικότερη αξιοποίηση τους σε αρκετούς τομείς. Κυβερνήσεις, δημόσιοι φορείς και εταιρίες πλέον αξιοποιούν και αναλύουν τα δεδομένα που διαθέτουν για να παρέχουν νέες και πιο εξελιγμένες υπηρεσίες προς πολίτες και καταναλωτές. Τα μεγάλα δεδομένα πλέον αναδιαμορφώνουν τις πολιτικές οργανισμών, προϊόντα προς ανάπτυξη και αποφάσεις. Τομείς όπως η υγεία, η παιδεία και η οικονομία χρησιμοποιούν πλέον μεγάλο όγκο δεδομένων για να εξάγουν γνώση.

2.2 Οι απαρχές της έννοιας των Μεγάλων Δεδομένων

Όπως προαναφέρθηκε ο όρος Big Data πρωτοεμφανίστηκε το 1998, ωστόσο η ανησυχία των μηχανικών ηλεκτρονικών υπολογιστών για το πως θα είναι δυνατόν να διαχειρίζονται μεγάλο όγκο δεδομένων υπάρχει από την δεκαετία του 80’. Τότε πολλοί οργανισμοί και εταιρίες εγκαθιστούσαν τα πρώτα τους πληροφοριακά συστήματα για να μειώσουν τις επιχειρησιακές δαπάνες ενώ ταυτόχρονα περνούσαν στην ψηφιακή εποχή. Σε αυτά τα συστήματα θα διατηρούσαν δεδομένα τα οποία με την πάροδο του χρόνου θα αυξάνονταν, και θα άλλαζαν μορφές.

Σήμερα αυτά τα πληροφοριακά συστήματα έχουν αντικατασταθεί από νέα, οι βάσεις δεδομένων έχουν εξελιχθεί και εργαλεία βελτιστοποιημένα ως προς την επεξεργασία και διαχείριση μεγάλων όγκων δεδομένων έχουν δημιουργηθεί. Ο όγκος των δεδομένων έχει αυξηθεί κατακόρυφα. Κάποιοι κάνουν λόγο ότι ο συνολικός αριθμός των δεδομένων που θα παραχθούν μέχρι το 2020 θα φτάσει τα 44 zetabytes. Για αυτόν τον λόγο, παρά τα βήματα που έχουν γίνει μέχρι και σήμερα, υπάρχει τεράστια ανάγκη για την επέκταση και βελτιστοποίηση υπαρχόντων τεχνολογιών ή εύρεση νέων.

2.3 Χαρακτηριστικά και προκλήσεις

Όπως έχει προαναφερθεί τα χαρακτηριστικά των μεγάλων δεδομένων μπορούν να κατηγοριοποιηθούν σε 3 κατηγορίες γνωστές και ως 3V. Ωστόσο πλεον κάποιοι υποστηρίζουν ότι υπάρχουν και άλλα παρεμφερή χαρακτηριστικά όπως το Value, το Validity, το Veracity και το Variability.

2.3.1 Volume

Το κύριο χαρακτηριστικό των μεγάλων δεδομένων είναι ο όγκος (Volume). Δεν θα μπορούσε να μην είναι ένα από τα βασικά χαρακτηριστικά τους και ίσως το σημαντικότερο. Χωρίς μεγάλο αριθμό εγγράφων σε file systems , δεδομένων σε βάσεις δεν θα μπορούσε να γίνεται λόγος για μεγάλα δεδομένα. Άρα και η λογική της οποιασδήποτε ανάλυσης θα κυμαινόταν στα πλαίσια της δειγματοληψίας και όχι του μεγάλου όγκου. Όσο μεγαλύτερος είναι ο αριθμός των δεδομένων τόσο πιο αντιπροσωπευτικό γίνεται το σύνολο των δεδομένων για ένα φαινόμενο. Ωστόσο όσο μεγαλύτερος είναι ο αριθμός των δεδομένων τόσο πιο δύσκολη και αργή γίνεται η επεξεργασία τους.

2.3.2 Velocity

Ένα δεύτερο χαρακτηριστικό είναι η ταχύτητα (Velocity). Η ταχύτητα χαρακτηρίζει τα δεδομένα πραγματικού χρόνου. Οι ταχύτητες ποικίλουν αναλόγως την ροή και μπορούν να μεταβάλλονται σε όγκο αλλά και σε ταχύτητα. Υπάρχουν πολλές εφαρμογές οι οποίες αναλαμβάνουν να διαχειριστούν δεδομένα πραγματικού χρόνου. Μερικές φορές η ταχύτητα με την οποία συλλέγονται τα δεδομένα είναι κατά πολύ μεγαλύτερη από την επεξεργαστική ταχύτητα των αλγορίθμων ή και των ίδιων των υλικών που υπάρχουν. Σε αυτές τις περιπτώσεις αναγκαστικά ένας αριθμός των συλλεγμένων δεδομένων απορρίπτεται και δεν χρησιμοποιείται στην ανάλυση. Ένα γνωστό παράδειγμα είναι η αναλυτική των δεδομένων που συλλέγονται κατά την διάρκεια λειτουργίας του επιταχυντή LHC στο CERN. Στο [3] αναφέρεται ότι αρκεί ένα δευτερόλεπτο λειτουργίας ώστε να παραχθούν 1 petabytes δεδομένων από τον LHC. Είναι λοιπόν φυσικό ακόλουθο να μην είναι εφικτή η αποθήκευση και η επεξεργασία τέτοιου όγκου δεδομένων με τόσο μεγάλη ταχύτητα.

2.3.3 Variety

Είναι γνωστό ότι τα δεδομένα μπορούν να λάβουν πολλές μορφές. Κείμενο, εικόνα, ήχος είναι μόλις μερικές αναπαραστάσεις των δεδομένων. Αυτά με την σειρά τους μπορούν να έχουν πολλούς τύπους (formats), μπορούν να είναι δομημένα ή μη δομημένα, επεξεργασμένα ή ακατέργαστα, σε αρχεία κάποιου file system, σε βάσεις δεδομένων, σε data warehouses, σε streams. Έτσι και τα μεγάλα δεδομένα ακολουθούν αυτόν τον κανόνα. Μπορεί να είναι οποιουδήποτε τύπου και μορφής. Πολλές φορές γίνονται προσπάθειες ώστε ετερογενή δεδομένα, όπως αυτά που προαναφέρθηκαν, να συνδυαστούν σε κάποια ανάλυση.

2.4 Διαχείριση μεγάλων δεδομένων

Όλα τα παραπάνω χαρακτηριστικά επηρεάζουν την διαχείριση που μπορούν να υποστούν τα δεδομένα και αυξάνουν την πολυπλοκότητα των αναλύσεων. Δεδομένα όπως τα posts του twitter, τα λεγόμενα tweets, μπορεί να είναι απλά αλφαριθμητικά δομημένα δεδομένα ωστόσο η ροή τους είναι αρκετά μεγάλη για να επεξεργαστούν σε πραγματικό χρόνο χωρίς τα απαραίτητα εργαλεία και αρκετή επεξεργαστική ισχύ. Από την άλλη δεδομένα που βρίσκονται σε αρχεία και βάσεις δεδομένων μπορεί να είναι μεγαλύτερης κλίμακας και πολυπλοκότητας.

Από τα παραπάνω συμπεραίνεται ότι η φύση των δεδομένων θα είναι εκείνη που θα καθορίσει το πως τελικά θα δομηθεί ένα σύστημα ικανό να επεξεργαστεί και να αναλύσει τα δεδομένα. Άρα με τον όρο διαχείριση μεγάλων δεδομένων (Big Data Management) εννοείται η διαδικασία κατά την οποία μεγάλες ποσότητες δομημένων ή αδόμητων δεδομένων οργανώνονται, διαμορφώνονται / μετασχηματίζονται και επιβλέπονται με στόχο την διασφάλιση της ποιότητας των δεδομένων και της χρησιμοποίησης τους από εφαρμογές ανάλυσης και επιχειρησιακής νοημοσύνης.

Παραδοσιακοί τρόποι επεξεργασίας δεν θα μπορούσαν να επιτύχουν άξια αποτελέσματα σε ένα τόσο μεγάλο όγκο δεδομένων. Έτσι οι απλές σχεσιακές βάσεις δεδομένων εξελίχθηκαν σε παράλληλες και μη-σχεσιακές οι οποίες προσφέρουν καλύτερο χρόνο απόκρισης θυσιάζοντας πολλές φορές ένα μέρος της αξιοπιστίας των δεδομένων που παρέχουν. Δημιουργήθηκαν εργαλεία και αρχιτεκτονικές παράλληλης επεξεργασίας δεδομένων όπως το δημοφιλές MapReduce [4], το οποίο χρησιμοποιείται ακόμα και σήμερα.

Όλο και περισσότερα εργαλεία και πλατφόρμες διαχείρισης μεγάλων δεδομένων δημιουργούνται κάθε χρόνο, με το Apache Hadoop [5] να είναι ένα από τα πρώτα που εμφανίστηκε χρησιμοποιώντας το MapReduce. Το συγκεκριμένο κάνει χρήση του HDFS (Hadoop Distributed File System) το οποίο πρόκειται για ένα κατανεμημένο file system που χρησιμοποιείται σε αρκετές πλατφόρμες διαχείρισης μεγάλων δεδομένων και το οποίο βασίστηκε στο Google File System [6]. Ένα άλλο συστατικό του Apache Hadoop είναι το *Hadoop YARN*, μία πλατφόρμα υπεύθυνη για την διαχείριση των πόρων ενός cluster κατά την διάρκεια της εκτέλεσης μίας πράξης πάνω στα δεδομένα. Προφανώς δεν μπορούν να λείπουν οι βιβλιοθήκες του Hadoop MapReduce.

Σε αυτήν την αρχιτεκτονική έχουν βασιστεί και μεταγενέστερες πλατφόρμες όπως το Apache Spark [7] το οποίο είναι αρκετές φορές πιο γρήγορο και μπορεί να εκτελέσει in memory πράξεις στα δεδομένα. Με παρόμοιο τρόπο διαχειρίζεται τα δεδομένα επίσης το Apache Storm [8] με την μόνη διαφορά ότι αντλεί δεδομένα μόνο από streams. Σε αυτήν την εργασία θα δοθεί έμφαση στο Apache Spark το οποίο χρησιμοποιήθηκε για την ανάλυση των δεδομένων.

Άλλη μία κατηγορία πλατφορμών που είναι άξια αναφοράς είναι τα νευρωνικά δίκτυα. Το όνομα οφείλεται στην αρχιτεκτονική η οποία υπάρχει σε αυτά τα δίκτυα

τα οποία λειτουργούν σαν βιολογικά νεύρα. Τέτοια συστήματα χρησιμοποιούνται κατά κόρον για μηχανική μάθηση (Machine Learning) το οποίο έχει ως στόχο να εκπαιδεύσει έναν έξυπνο αλγόριθμο για να βγάζει αποτελέσματα τα οποία έχουν κάποια αξία ή κάνουν κάποια πρόβλεψη.

2.5 Μεγάλα Δεδομένα και Αναλυτική

Τα δεδομένα εξορισμού είναι μία συλλογή από τιμές που περιγράφουν καταστάσεις αντικειμένων, προσώπων, φαινομένων. Από αυτό γίνεται αντιληπτό ότι ο τύπος και τα κριτήρια με τα οποία επιλέγονται τα δεδομένα μπορεί να διαφέρει στην κάθε μια από τις παραπάνω κατηγορίες. Όταν τα δεδομένα περιγράφουν την κατάσταση ενός ανθρώπου είναι πιο εύστοχο να κρατούνται δεδομένα όπως η ηλικία, το όνομα, η οικογενειακή κατάσταση, ενώ για δεδομένα που περιγράφουν ένα αντικείμενο θα έχει νόημα το χρώμα, η τιμή, η μάρκα, ο αριθμός των υπολοίπων σε κάποια αποθήκη. Αλλά και πάλι αυτό μπορεί να γίνει σχετικό. Ένας οργανισμός μπορεί να χρειάζεται να γνωρίζει το εισόδημα ενός ατόμου αλλά ένας άλλος να προτιμά μόνο τις καταναλωτικές του συνήθειες.

Αυτή η εργασία εστιάζει στα χώρο-χρονικά δεδομένα και στις τροχιές, δηλαδή δεδομένα που περιγράφουν ένα φαινόμενο στις διαστάσεις του χώρου και του χρόνου με στατικό τρόπο και καθώς αυτό εξελίσσεται αντίστοιχα. Αλλά πιο πριν θα εξεταστεί η κάθε διάσταση ξεχωριστά ώστε η μετάβαση για τον αναγνώστη να γίνει σταδιακά.

2.5.1 Χωρικά δεδομένα

Τα χωρικά δεδομένα (Spatial Data) αναφέρονται σε όλους εκείνους τους τύπους δεδομένων που μπορούν να περιγράψουν μία γεωγραφική τοποθεσία στο επίπεδο ή στο χώρο. Συνήθως αποτυπώνονται ως συντεταγμένες σε ένα γεωγραφικό σύστημα συντεταγμένων και με την βοήθεια ενός GIS (Geographic Information System) εργαλείου μπορούν να επεξεργαστούν, να αναλυθούν και να αναπαρασταθούν σε μορφές κατανοητές για τους ανθρώπους.

Χωρικά δεδομένα παράγονται καθημερινά από τον κάθε άνθρωπο που μεταφέρει ένα κινητό τηλέφωνο, είτε μέσω του GPS, είτε μέσω των σημάτων που στέλνονται στις κυψέλες των εταιριών κινητής τηλεφωνίας. Ανά πάσα στιγμή κάποιος μπορεί να βρει το στίγμα του σε έναν χάρτη και να κατευθυνθεί οπουδήποτε θελήσει. Όλα αυτά με την βοήθεια της ανάλυσης των χωρικών δεδομένων που έχουν συλλεχθεί. Τέτοιες μορφές δεδομένων μπορούν να χρησιμοποιηθούν και από οργανισμούς για κρίσιμους σκοπούς.

Για να γίνει αντιληπτό το πόσο σημαντικά και κρίσιμα μπορεί να είναι τα χωρικά δεδομένα αρκεί να ειπωθεί ότι έχουν γίνει μεγάλες επενδύσεις στον τομέα αυτόν. Παραδείγματα υπάρχουν πολλά. Ένα από αυτά είναι η δημιουργία του γνωστού σε

όλους GPS του οποίου η λειτουργία απαιτεί έναν σεβαστό αριθμό δορυφόρων σε τροχιά γύρω από την γη. Ένα ακόμα, απόρροια της τεράστιας πληροφορίας που παράγεται, είναι οι χωρικές βάσεις δεδομένων οι οποίες βελτιστοποιούν την αποθήκευση και αναζήτηση τέτοιων τύπων δεδομένων.

Η ανάλυση των χωρικών δεδομένων μπορεί να γίνει σχετικά εύκολα θεωρώντας ένα πλέγμα κελιών (Grid) πάνω από έναν χάρτη. Οποιαδήποτε παρατήρηση εμφανίζεται μέσα σε ένα κελί (Cell) θεωρείται ότι ανήκει μόνο σε αυτό και όταν φτάσει η ώρα της ανάλυσης θα προσμετρηθεί στην σημαντικότητα του κελιού. Τα αποτελέσματα που εξάγονται από μία τέτοια ανάλυση εξαρτώνται από το μέγεθος που θα δώσει ο ερευνητής στο κάθε κελί του πλέγματος αλλά και τα στατιστικά που θα χρησιμοποιήσει.

2.5.2 Χρονικά δεδομένα

Το κύριο χαρακτηριστικό των χρονικών δεδομένων είναι ο χρόνος, ο οποίος αποτυπώνεται με την μορφή ώρας ή ημερομηνίας. Μερικές φορές είναι πιο εύστοχο να παρατηρηθεί ένα φαινόμενο στο πως εξελίσσεται κατά την πάροδο του χρόνου. Για παράδειγμα η κίνηση σε ένα συγκεκριμένο κόμβο της πόλης, ώστε μετά από κάποια επεξεργασία και ανάλυση να μπορεί να ρυθμιστεί η κυκλοφορία των οχημάτων με ομαλότερο τρόπο. Και σε αυτή την κατηγορία υπάρχουν πληθώρα δεδομένων τα οποία παράγονται ανά πάσα στιγμή.

Η ανάλυση των χρονικών δεδομένων είναι μία μεγάλη πρόκληση καθώς στις περισσότερες εφαρμογές πρέπει να αναλύονται σε πραγματικό χρόνο. Όπως προαναφέρθηκε μερικές φορές η ροή της πληροφορία που συλλέγεται γίνεται αρκετά μεγαλύτερη από την ταχύτητα επεξεργασίας των συστημάτων που την αναλύουν. Τέλος, είναι άξιο αναφοράς ότι υπάρχουν αφοσιωμένες βάσεις δεδομένων σε χρονικά δεδομένα.

2.5.3 Χώρο-χρονικά δεδομένα

Τα χώρο-χρονικά δεδομένα είναι ο συνδυασμός των χρονικών και χωρικών δεδομένων, δηλαδή μπορούν να αναπαρασταθούν σε 4 διαστάσεις (μήκος, πλάτος, ύψος, χρόνος). Συνήθως αποτυπώνουν παρατηρήσεις αντικειμένων με αυτά τα χαρακτηριστικά, μία γεωγραφική απεικόνιση του αντικειμένου σε ένα GIS και ο χρόνος παρατήρησης αυτού. Ένα παράδειγμα μπορεί να είναι οι αποβιβάσεις πελατών ταξί σε μία πόλη όπως μελετήθηκαν στα πλαίσια του διαγωνισμού GISCup 2016 [20] . Αξίζει να αναφερθεί ότι ο αλγόριθμος BigCAB, ο οποίος θα παρουσιαστεί σε επόμενο κεφάλαιο, δημιουργήθηκε για να αναλύει χώρο-χρονικά δεδομένα και πιο συγκεκριμένα για να ανιχνεύει γεωγραφικά τμήματα με μεγάλη στατιστική σημαντικότητα σε κάποιο χρονικό πλαίσιο.

2.5.4 Δεδομένα τροχιών

Άλλος ένας παρεμφερής αλλά πιο σύνθετος όρος που χρησιμοποιείται είναι η τροχιές (Trajectories). Αν και έχει γίνει μελέτη από πολλούς ερευνητές για δεδομένα τροχιών υπάρχουν ακόμα αρκετές δυσκολίες στην αποτύπωση, επεξεργασία και ανάλυση αυτών.

Υπάρχουν διαφορετικές προσεγγίσεις ως προς την αποτύπωση των δεδομένων αναλόγως την τεχνολογία που χρησιμοποιείται και τα αντικείμενα τα οποία παρατηρούνται όπως γράφεται στο [9]. Και αυτό ισχύει αν αναλογιστεί κανείς την ευρύτητα που μπορούν να έχουν τα χώρο-χρονικά δεδομένα. Σίγουρα χρειάζονται οι συντεταγμένες στις οποίες παρατηρήθηκε το αντικείμενο και ο χρόνος παρατήρησης για να θεωρηθεί κάποιο σύνολο από δεδομένα, χώρο-χρονικό. Από κει και έπειτα μπορεί κάποιος να θεωρήσει οποιαδήποτε άλλη μεταβλητή η στοιχείο που μπορεί να θέλει να μελετήσει όπως ταχύτητα, κατεύθυνση και άλλα, τα οποία περιγράφουν την κατάσταση ενός αντικειμένου εκείνη την χρονική στιγμή.

Άρα αυτά τα δεδομένα μπορούν να εκφραστούν ως μία ακολουθία σημείων/παρατηρήσεων ενός κινούμενου αντικειμένου τα οποία συνδέονται μεταξύ τους με κάποιο μοναδικό αναγνωριστικό και διαθέτουν πληροφορίες για τουλάχιστον την τοποθεσία στην οποία παρατηρήθηκαν την χρονική στιγμή που παρατηρήθηκαν.

3. Εργαλεία ανάλυσης μεγάλων δεδομένων

3.1 Εισαγωγή

Σε μία εποχή στην οποία οι πληροφορίες παράγονται με μεγάλη ταχύτητα και όγκο κάθε διαθέσιμο δεδομένο θα μπορούσε να χρησιμοποιηθεί για την παραγωγή γνώσης. Πολλές φορές η γνώση που παράγεται από τα δεδομένα είναι καίριας σημασίας για έναν οργανισμό. Έτσι όλο και περισσότεροι οργανισμοί άρχισαν να συλλέγουν δεδομένα και να επιζητούν να κερδίσουν μέσα από αυτά. Αυτό οδήγησε στην εξέλιξη λογισμικών, αφοσιωμένων στην ανάλυση των μεγάλων δεδομένων.

Αυτό το κεφάλαιο είναι αφιερωμένο σε τέτοια λογισμικά και πλατφόρμες. Αρχικά γίνεται λόγος για το μοντέλο του MapReduce το οποίο πρόκειται για το δημοφιλέστερο μοντέλο ανάλυσης δεδομένων και έπειτα γίνεται αναφορά στο Apache Hadoop. Το κεφάλαιο κλείνει με περιγραφή της αρχιτεκτονικής, του τρόπου λειτουργίας αλλά και διαχείρισης των δεδομένων του Apache Spark.

3.2 MapReduce

Όπως προαναφέρθηκε το MapReduce είναι ένα προγραμματιστικό μοντέλο για παράλληλη επεξεργασία μεγάλων δεδομένων σε κατανεμημένο περιβάλλον. Τα βασικά συστατικά του, όπως αναφέρεται και στο όνομα είναι δύο διακριτές διαδικασίες, η Map και η Reduce. Υπάρχει ακόμα μία διαδικασία ανάμεσα στις δύο πρώτες η οποία ονομάζεται Shuffle. Η επεξεργασία των δεδομένων γίνεται εξολοκλήρου από τους επιμέρους κόμβους ενός cluster. Αυτοί ονομάζονται και Worker Nodes. Επίσης υπάρχει ένας κόμβος ο οποίος επιβλέπει αυτή την διαδικασία ο οποίος ονομάζεται Master Node. Αυτός εκκινεί την διαδικασία της επεξεργασίας και διατάσσει τα Worker Nodes.

3.2.1 Map

Η διαδικασία του Mapping μπορεί να προσφέρει μία πρώτη επεξεργασία στα δεδομένα. Αυτή η επεξεργασία μπορεί να περιλαμβάνει μετασχηματισμούς και φιλτράρισμα. Η τελική μορφή των δεδομένων μετά από αυτή την διαδικασία θα είναι ζευγάρια από κλειδιά και τιμές (key-value pairs). Για παράδειγμα αν κάποιος θέλει να

βρει τις πιο συχνά εμφανιζόμενες λέξεις σε ένα κείμενο η συνάρτηση Map που θα χρησιμοποιήσει θα είναι η εξής:

```
Map(Document d):
    for each word w in d:
        return pair(w, 1);
```

Δηλαδή για κάθε λέξη w μέσα στο έγγραφο d θα επιστραφεί ένα νέο key-value pair με κλειδί την λέξη w και τιμή τον αριθμό 1. Έτσι στην ουσία κάθε κόμβος του cluster θα πάρει ένα μέρος του εγγράφου και θα το κατακερματίσει σε διακριτές λέξεις. Ο Master Node σε αυτό το σημείο διασφαλίζει ότι κάθε σημείο του εγγράφου θα πάει μόνο σε ένα Worker Node. Επίσης σε περίπτωση σφάλματος ή αστοχίας ενός Worker Node θα είναι εκείνος που θα ξαναστείλει το προς επεξεργασία κομμάτι που χάθηκε σε κάποιο άλλο.

3.2.2 Shuffle

Η διαδικασία του shuffling των δεδομένων είναι κρίσιμης σημασίας στο MapReduce μοντέλο. Είναι αυτή που θα καθορίσει τελικά το πως θα διαμοιραστούν τα δεδομένα στα επιμέρους nodes ενός cluster για την τελική φάση του Reduce. Στόχος αυτού του βήματος είναι να πάνε όλα τα key-value pairs με το ίδιο κλειδί στο ίδιο Worker Node για την φάση του Reduce. Η διαδικασία του shuffling γίνεται αυτόματα από τα frameworks που υποστηρίζουν το μοντέλο του MapReduce και ο προγραμματιστής δεν μπορεί να επέμβει.

3.2.1 Reduce

Το reduce είναι η διαδικασία κατά την οποία τα δεδομένα με ίδια κλειδιά έχουν σταλεί σε ένα Worker Node το οποίο θα τα επεξεργαστεί με όμοιο τρόπο. Στο παραπάνω παράδειγμα με τις πιο συχνά εμφανιζόμενες λέξεις ενός κειμένου η υλοποίηση του Reduce θα ήταν απλή:

```
Reduce(p1, p2):
    return p1.value + p2.value;
```

Δηλαδή για κάθε key-value pair τα οποία έχουν ίδιο κλειδί, θα επιστραφεί το άθροισμα των τιμών τους.

3.3 Apache Hadoop

Το Apache Hadoop ήταν η πρώτη επιτυχημένη πλατφόρμα που χρησιμοποίησε το μοντέλο MapReduce. Η μεγάλη της επιτυχία οφειλόταν στο ότι μείωσε αρκετά τους

χρόνους ανάλυσης δεδομένων ενώ ταυτόχρονα κατέβασε την πολυπλοκότητα της υλοποίησης των αλγορίθμων επεξεργασίας δεδομένων. Ένα ακόμα κλειδί που συντέλεσε στη φήμη που απέκτησε το Hadoop είναι το HDFS (Hadoop Distributed File System), το οποίο πρόκειται για ένα κατανεμημένο file system όπου μοιράζονται όλα τα Nodes ενός cluster. Αυτή είναι και η κύρια διαφορά από το παραδοσιακό μοντέλο του MapReduce. Ο κάθε κόμβος διαθέτει ένα κομμάτι των δεδομένων καθώς διαμοιράζονται σε όλα τα Nodes μέσω του HDFS. Έτσι η πληροφορία είναι ήδη στους κόμβους και δεν χρειάζεται κάποιο αρχικό partition πριν την εκκίνηση της επεξεργασίας των δεδομένων. Σε αυτό το σημείο θα γίνει εκτενέστερη αναφορά στο HDFS καθώς παίζει καθοριστικό ρόλο στο Apache Spark.

Το HDFS είναι σχεδιασμένο ώστε να κατανέμει και να αποθηκεύει μεγάλο όγκο δεδομένων με τρόπο ισόποσο και αποδοτικό σε όλους τους κόμβους που το αποτελούν. Αυτό από μόνο του είναι αρκετά αποδοτικό καθώς τα δεδομένα είναι ήδη διαθέσιμα στους κόμβους για επεξεργασία. Όπως γίνεται κατανοητό αυτό μειώνει την άσκοπη αποστολή δεδομένων μέσα στο δίκτυο του cluster και αυξάνει σημαντικά την απόδοση του συστήματος. Είναι άξιο να αναφερθεί ότι τα αρχεία τεμαχίζονται σε ισόποσα κομμάτια τα οποία ονομάζονται blocks για την καλύτερη κατανομή τους μέσα στο file system.

Επίσης το HDFS είναι σχεδιασμένο έτσι ώστε να θεωρεί την αστοχία υλικού συχνό φαινόμενο. Έτσι φροντίζει να δημιουργεί αντίγραφα των δεδομένων ενός κόμβου σε έναν άλλον για αποφυγή απώλειας δεδομένων. Ο αριθμός των αντιγράφων ορίζεται από μία μεταβλητή η οποία ονομάζεται Replication Factor. Όταν ο κόμβος επανέλθει αρκεί ένα rebalancing για να κατανεμηθούν τα δεδομένα και πάλι ομοιόμορφα και να επαναδημιουργηθούν αντίγραφα ασφαλείας εάν η κατάσταση το επιτάσσει.

Το HDFS ως κατανεμημένο σύστημα ακολουθεί την λογική του Master/Slave που χρησιμοποιεί και το MapReduce. Εδώ ο MasterNode ονομάζεται NameNode και πρόκειται για έναν κεντρικό κόμβο ο οποίος αναλαμβάνει την διαχείριση του συστήματος. Οι υπόλοιποι κόμβοι υπό τον NameNode ονομάζονται DataNodes. Η διαχείριση έχει να κάνει με την κατανομή των δεδομένων σε όλους τους κόμβους, η δημιουργία αντιγράφων και ο έλεγχος της προσπέλασης αυτών από τις εφαρμογές που θα τρέξουν από πάνω (Hadoop, Spark κτλ). Επίσης διατηρεί πληροφορίες για το Namespace του file system και μπορεί να διατάσσει τα DataNodes να προχωρήσουν σε πράξεις πάνω στα αρχεία.

3.4 Apache Spark

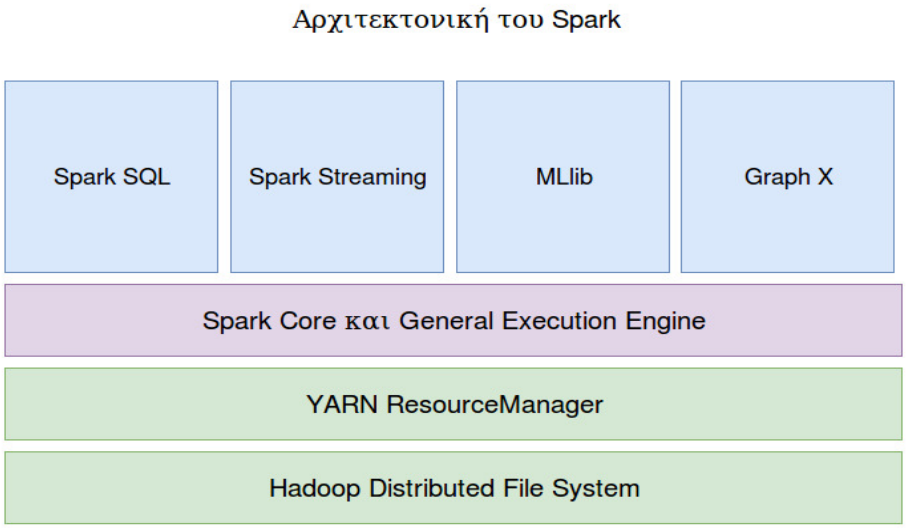
Το Apache Spark πρόκειται για ένα δημοφιλές εργαλείο ανάλυσης δεδομένων. Υποστηρίζει αρκετές γλώσσες προγραμματισμού όπως η Java, η Scala και η Python. Είναι ίσως το πιο γρήγορο εργαλείο ανοιχτού κώδικα για ανάλυση δεδομένων από αρχεία και βάσεις δεδομένων. Χρησιμοποιεί το MapReduce για την αποτελεσματική εκτέλεση παράλληλων εργασιών (task) και θεωρείται ως μία από τις καλύτερες λύσεις για επεξεργασία μεγάλου όγκου δεδομένων. Το Spark χρησιμοποιεί τα Resilient Distributed Datasets (RDDs) [12] τα οποία επιτρέπουν πράξεις μεταξύ

δεδομένων απευθείας στην μνήμη των επιμέρους κόμβων ενός cluster. Κάθε task που δίνεται στο Spark αποτελεί είτε μία δημιουργία ενός RDD, είτε μια τροποποίηση (Transformation), είτε μία πράξη (Action) πάνω σε ένα RDD. Επιπλέον το spark χρησιμοποιεί το HDFS για την αποθήκευση και κατανομή των αρχείων δεδομένων στους κόμβους.

3.4.1 Αρχιτεκτονική Spark

Η αρχιτεκτονική που ακολουθεί το Spark δεν διαφέρει από αυτή που χρησιμοποίησαν άλλα εργαλεία όπως το προαναφερθέν Apache Hadoop. Εξάλλου θεωρείται και αυτό ένα κομμάτι του οικοσυστήματος που έχει δημιουργήσει μαζί με πληθώρα άλλων εργαλείων. Έτσι και αυτό χρησιμοποιεί το HDFS σαν κύρια πηγή δεδομένων ωστόσο μπορεί να προσαρμοστεί και σε άλλες, όπως η Hbase [17], η Cassandra [18] και οποιαδήποτε άλλη πηγή πληροφοριών του οικοσυστήματος του Hadoop, με τις απαραίτητες παραμετροποιήσεις.

Το Spark και αυτό μπορεί να χρησιμοποιήσει το Yarn ή το Mesos [19] ως μέσα επίβλεψης των εφαρμογών και των πόρων στους κόμβους του cluser. Όμως δεν είναι απαραίτητη η ύπαρξη τους καθώς το Spark διαθέτει ήδη Resource και Application Manager. Ένας Resource Manager είναι εκείνος ο οποίος θα δεσμεύσει πόρους στους κόμβους του cluster ώστε η οποιαδήποτε εφαρμογή που θα χειριστεί από τον Application Manager να εκτελεστεί με επιτυχία. Αυτά τα εργαλεία λειτουργούν ως διαμεσολαβητές μεταξύ του client που θα εκκινήσει την εφαρμογή και των επιμέρους κόμβων που θα αναλάβουν την επεξεργασία των δεδομένων. Αξίζει να αναφερθεί ότι η υλοποίηση σε αυτή την εργασία βασίστηκε στο Spark over Yarn μοντέλο.



Εικόνα 1: Αρχιτεκτονική Apache Spark και βιβλιοθήκες

Συνεχίζοντας υπάρχει το επίπεδο της εφαρμογής του Spark. Αυτό αποτελείται από 5 βιβλιοθήκες. Η πιο σημαντική και αυτή που διαθέτει τις απαραίτητες κλάσεις για να τρέξει μία εφαρμογή στο Spark είναι το Spark Code. Η κεντρική βιβλιοθήκη του Spark στην οποία βασίζονται και οι άλλες. Άλλες βιβλιοθήκες είναι η Spark SQL που εμπεριέχει μηχανισμούς διασύνδεσης με βάσεις τόσο σχεσιακές όσο και μη σχεσιακές, η Spark Streaming η οποία διαχειρίζεται ροές από δεδομένα πραγματικού χρόνου. Επίσης υπάρχει η MLlib, υπεύθυνη για εφαρμογές εκμάθησης μηχανής και η GraphX η οποία χρησιμοποιείται για επεξεργασία γράφων.

3.4.2 Resilient Distributed Datasets και RDD Operations

Τα RDDs είναι ανεκτικές σε σφάλματα, παράλληλες δομές δεδομένων, που επιτρέπουν στον χρήστη να κρατήσει ενδιάμεσα αποτελέσματα στην μνήμη, να ελέγξει τον διαμερισμό τους ώστε να κατανέμει τα δεδομένα με τον βέλτιστο τρόπο και να τα επεξεργαστεί αποτελεσματικά χρησιμοποιώντας μια πλούσια εργαλειοθήκη από πράξεις όπως αναφέρεται στο [12]. Τα RDDs είναι το μέσο που χρησιμοποιεί το Spark για να δομήσει την πληροφορία που αντλεί από τις πηγές του. Κάποιος θα μπορούσε να το παρομοιάσει με έναν συμβολικό σύνδεσμο (Symbolic Link) μεταξύ του προγράμματος που τρέχει και των δεδομένων που είναι διαμοιρασμένα σε έναν HDFS ή στην μνήμη ενός συστήματος.

Αυτές οι δομές δεδομένων μπορούν να δημιουργηθούν είτε από πηγές δεδομένων όπως βάσεις και HDFS, είτε από άλλα RDDs. Επίσης αξίζει να αναφερθεί ότι αυτές οι δομές δεδομένων είναι μη επανεγγράψιμες οπότε κάθε μετασχηματισμός που γίνεται πάνω στα ήδη διαθέσιμα στοιχεία ενός RDD δημιουργεί νέα στοιχεία.

Όταν κάποιος δηλώσει ένα RDD να διαβάσει μία πηγή δεδομένων, δηλαδή να εκτελέσει ένα Map, η διαδικασία δεν θα ξεκινήσει απευθείας. Όπως ειπώθηκε παραπάνω το RDD είναι κάτι σαν συμβολικός σύνδεσμος προς τα δεδομένα. Γνωρίζει που είναι ωστόσο δεν τα έχει φορτώσει ακόμα στην μνήμη για να ξεκινήσει να τα επεξεργάζεται. Αυτό γιατί τα RDDs θα ξεκινήσουν την διαδικασία της επεξεργασίας μόνο όταν τα δεδομένα χρειαστούν. Το πότε τα δεδομένα θα χρειαστούν τελικά καθορίζεται από τις πράξεις που θα εκτελεστούν.

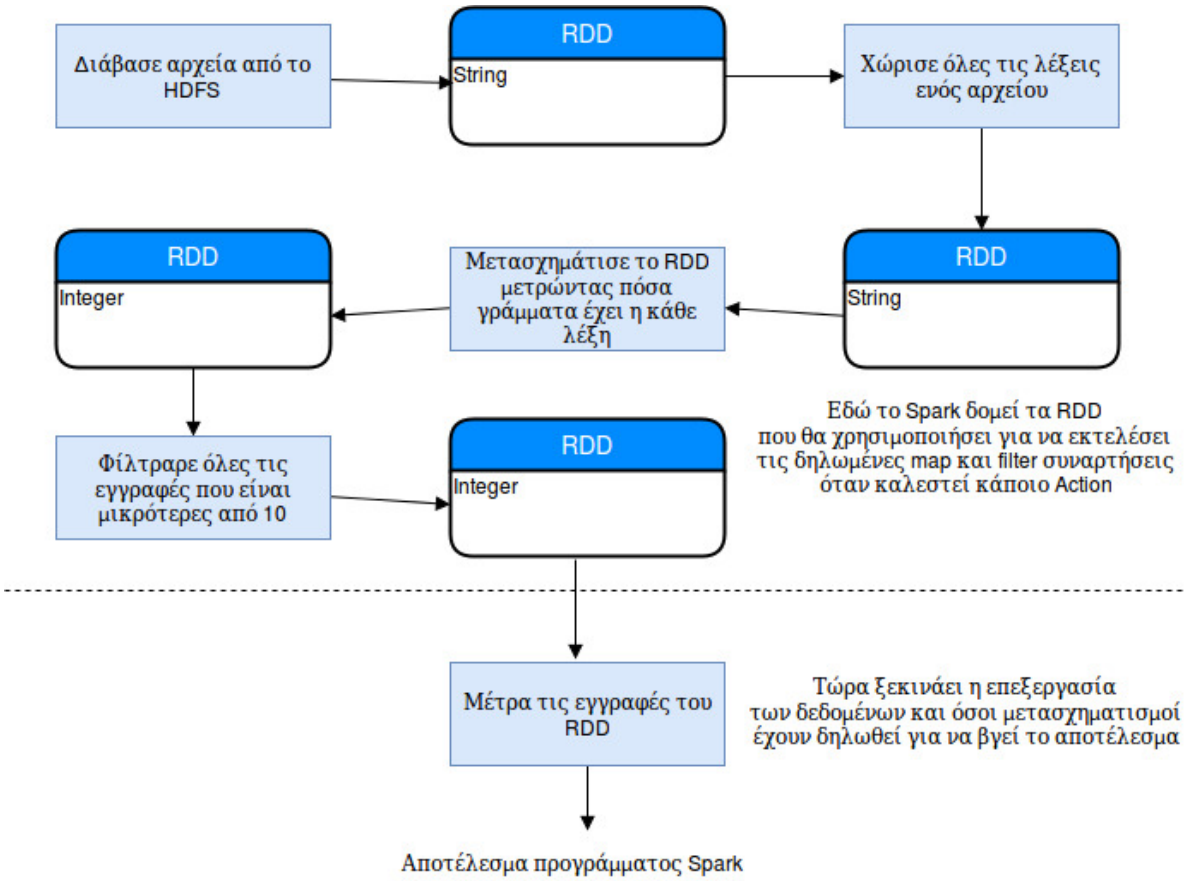
Πιο πριν είχαν αναφερθεί τα Transformations και τα Actions. Αυτά είναι τα δύο είδη των RDD Operations. Τα transformations κάνουν αλλαγές πάνω στα RDDs, δηλαδή αναδομούν τα δεδομένα σε μορφές που έχουν αξία για την επεξεργασία τους. Τέτοιες αλλαγές μπορούν να εκτελέσουν εντολές όπως το map, το reduceByKey, το filter, το join. Οπότε όσα από τα παραπάνω και αν δηλωθούν προς εκτέλεση, κανένα δεν πρόκειται πραγματικά να εκτελεστεί εάν δεν δηλωθεί κάποιο Action πάνω στο τελικό RDD.

Τα Actions είναι η δεύτερη μορφή operations που διαθέτουν τα RDDs. Είναι σημαντικό να γίνει αντιληπτό ότι τα Actions μπορούν να εφαρμοστούν μόνο πάνω σε RDDs και μπορούν να παράξουν μόνο κανονικά δεδομένα. Υπάρχουν αρκετές

συναρτήσεις για actions, μερικές από αυτές είναι το count, το first, το max, το min. Στην εικόνα που ακολουθεί φαίνεται ένα παράδειγμα μίας εκτέλεσης ενός προγράμματος Spark το οποίο ξεκινάει την εκτέλεση του όταν δηλωθεί και η τελευταία μέθοδος προς εκτέλεση.

Όπως φαίνεται αρχικά φτιάχνεται ένα RDD το οποίο διαβάζει αρχεία από το HDFS. Στην συνέχεια χωρίζοντας όλες τις λέξεις του κειμένου σε ξεχωριστές καταχωρίσεις δημιουργεί ένα νέο RDD. Έπειτα μετράει τα γράμματα κάθε καταχώρισης και δημιουργεί ένα νέο RDD, αυτή τη φορά από αριθμούς. Επιπλέον φιλτράρει όλες τις καταχωρίσεις που έχουν αριθμό μικρότερο από 10, δηλαδή όλες τις λέξεις που έχουν λιγότερα από 10 γράμματα. Το Spark μέχρι στιγμής όμως δεν θα έχει εκτελέσει τίποτα από αυτά πραγματικά. Αφού όλοι αυτοί αποτελούν μετασχηματισμούς και πράξεις πάνω σε RDDs οι οποίες δημιουργούν νέα RDD.

Στο τελευταίο βήμα ο αλγόριθμος φαίνεται να καλεί ένα Action το οποίο παράγει αποτέλεσμα διαφορετικό από RDD. Όταν προσπελαστεί αυτό το σημείο από την πλατφόρμα του Spark τότε θα ξεκινήσει η κανονική εκτέλεση όλων των προηγούμενων βημάτων.



Εικόνα 2: Παράδειγμα εκτέλεσης Transformations και Actions στο Spark

Παραδοσιακά όταν ξεκινήσει το Spark να επεξεργάζεται τα δεδομένα τα κρατάει στην μνήμη των κόμβων. Όταν ο όγκος των δεδομένων που υπάρχουν στην μνήμη φτάσουν κοντά στο όριο που έχει οριστεί κατά την εκκίνηση του προγράμματος, τότε γράφονται στον δίσκο. Τα RDDs προσφέρουν διάφορες στρατηγικές αποθήκευσης. Μπορούν να κρατηθούν εξολοκλήρου στην μνήμη ή στον δίσκο, μπορούν να κατανεμηθούν με αυτόματο και έξυπνο τρόπο σε όλους τους κόμβους. Επίσης μπορούν να δοθούν προτεραιότητες σε κάποια RDD εις βάρος άλλων οπότε και να είναι τα δεύτερα αυτά που θα γραφούν στον δίσκο σε περίπτωση που η μνήμη γεμίσει.

3.4.3 Yarn

Το Yarn είναι ένας διαχειριστής πόρων και υπεύθυνος για την εποπτεία και διαχείριση παράλληλων διεργασιών και χρησιμοποιείται κατά κόρον σε κατανεμημένα συστήματα και εφαρμογές οι οποίες επεξεργάζονται δεδομένα παράλληλα. Όπως διαπιστώνεται αποτελείται από δύο τμήματα. Αυτό του Scheduler και αυτό του Application Manager και η βασική ιδέα του Yarn είναι η κατάτμηση αυτών των δύο όρων.

Ο Scheduler είναι υπεύθυνος για την απόδοση των πόρων που θα του ζητηθούν από κάθε κόμβο του συστήματος. Είναι το μόνο που έχει αρμοδιότητα να κάνει. Η απόδοση των πόρων γίνεται βάσει των απαιτήσεων μίας εφαρμογής μέσω του Application Master. Ο Application Master είναι μία οντότητα η οποία είναι υπεύθυνη για την εκτέλεση μία εφαρμογής. Κάθε εφαρμογή που τρέχει σε ένα σύστημα έχει τον δικό της Application Master.

Κάθε κόμβος διαθέτει έναν Node Manager ο οποίος έχει αρμοδιότητα να εκτελέσει επιμέρους διεργασίες της εφαρμογής και να επιβλέψει την εξέλιξη αυτών. Για να ξεκινήσει ωστόσο η εκτέλεση μίας διεργασίας σε έναν κόμβο θα πρέπει ο Application Master να πάρει την άδεια του Applications Manager, του δεύτερου τμήματος του Resource Manager.

3.4.4 Spark on Yarn Αρχιτεκτονική

Στην Spark on Yarn αρχιτεκτονική υπάρχουν τρεις διακριτοί ρόλοι. Ο Spark Driver, ο Yarn Master και τα Spark Executors. Ο Spark Driver είναι αυτός που θα εκκινήσει την εφαρμογή και θα εκτελεί το κεντροποιημένο (centralized) κομμάτι της εφαρμογής. Ο Yarn Master αναλαμβάνει την δημιουργία και τον συντονισμό των Yarn Containers. Τα Yarn Containers είναι ταυτόχρονα και τα Spark Executors. Ο Master κόμβος επιλέγεται τυχαία όταν ξεκινάει η εφαρμογή από έναν από τους διαθέσιμους κόμβους του cluster. Ένα μέρος των υπόλοιπων (ή όλοι, αναλόγως πόσους έχει αιτηθεί η εφαρμογή) γίνονται οι Spark Executors.

Υπάρχουν δύο τρόποι λειτουργίας της Spark on Yarn αρχιτεκτονικής. Το Client Mode και το Cluster Mode και επηρεάζει το που θα εκτελεστεί ο Spark Driver. Στο Client Mode ορίζεται ως Spark Client ο κόμβος στον οποίον εκτελέστηκε η εντολή εκκίνησης της Spark εφαρμογής. Δηλαδή στην συγκεκριμένη περίπτωση ο ίδιος ο client που ζήτησε την εκτέλεση μίας εφαρμογής γίνεται ταυτόχρονα και Application Master. Στον δεύτερο τρόπο εκτέλεσης, το Cluster Mode, ο Spark Driver επιλέγεται τυχαία κατά την αρχικοποίηση της εφαρμογής. Ο προεπιλεγμένος τρόπος λειτουργίας που διαθέτει το Spark on Yarn σε περίπτωση όπου ο χρήστης δεν το δηλώσει είναι το Client Mode.

4. Hotspot analysis

4.1 Εισαγωγή

Αυτή η εργασία θα εστιάσει στο Hotspot Analysis και πιο συγκεκριμένα στο στατιστικό Getis-Ord G_i^* [10]. Το Hotspot Analysis πάνω σε χωρικά δεδομένα έχει στόχο την εύρεση των σημείων μεγάλης στατιστικής σημαντικότητας σε μία γεωγραφική έκταση. Όσο μεγαλύτερη είναι η τιμή ενός σημείου τόσο μεγαλύτερη είναι και η σημασία του για την ανάλυση. Πρόκειται για μία πρακτική ομαδοποίησης (Clustering) των δεδομένων που εμφανίζονται να έχουν τις ίδιες συνήθειες.

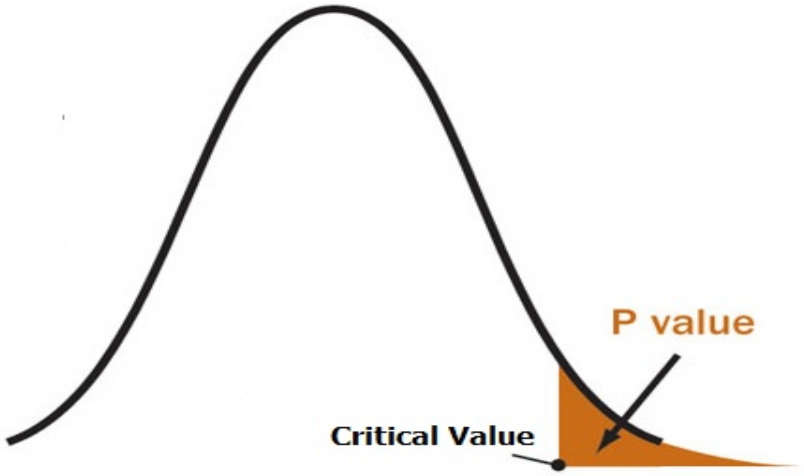
Αυτή η τεχνική χρησιμοποιείται κατά κόρον σε στατιστικές μελέτες που γίνονται καθημερινά. Αρκεί κανείς να δει διάφορες μελέτες εξάπλωσης ιών ή γρίπης, μελέτες που έχουν να κάνουν με κοινωνικά στοιχεία και άλλα παρόμοια. Παράλληλα κάποιος στατιστικός μπορεί να μελετήσει και τα σημεία των γεωγραφικών περιοχών που παρουσιάζουν μικρές τιμές τα οποία ονομάζονται Cold Spots. Σε κάθε περίπτωση η μελέτη των Hotspot παρέχει δύο στατιστικά αποτελέσματα. Το Z-Score και το P-value τα οποία θα εξηγηθούν στην συνέχεια.

Υπάρχουν δύο ευρέως διαδεδομένες μέθοδοι για Hotspot Analysis. Η προαναφερθείσα Getis-Ord G_i^* και η Moran’s I [11]. Στο κεφάλαιο που ακολουθεί εξηγούνται οι έννοιες του Z-Score και του P-Value και γίνεται μία αναφορά στο Moran’s I στατιστικό και αναλυτική περιγραφή του G_i^* στατιστικού ενώ. Επίσης παρουσιάζεται ένας τρόπος κατάτμησης του χώρου της ανάλυσης και αναφέρονται τρόποι Hotspot ανάλυσης σε διάφορους τύπους μεγάλων δεδομένων.

4.2 Z-Score και P-Value

Όπως προαναφέρθηκε το Hotspot Analysis έχει ως σκοπό την εύρεση σημείων μεγάλης στατιστικής σημαντικότητας. Η βαρύτητα της σημαντικότητας καθορίζεται από δύο τιμές το Z-Score και το P-value.

Το Z-Score, με απλά λόγια, είναι ο αριθμός των σταθερών αποκλίσεων που έχει μία ομάδα παρατηρήσεων από τον μέσο όρο στα πλαίσια της κανονικής κατανομής. Άρα όσο μεγαλύτερη ή μικρότερη από τον μέσο όρο είναι η τιμή του Z-Score τόση μεγαλή σημαντικότητα αποκτά αυτή η παρατήρηση. Στην ουσία δεν φτάνει μόνο η τιμή του Z-Score να είναι πολύ μεγάλη ή πολύ μικρή, αλλά να έχει μία αρκετά μεγάλη απόσταση από τον μέσο όρο της κατανομής.



Εικόνα 3: Αποτύπωση P-value στην κανονική κατανομή

Το P-value έρχεται να συμπληρώσει το Z-Score και αναφέρεται στην τυχαιότητα ενός γεγονότος ή μίας παρατήρησης. Όσο πιο μεγάλο είναι το P-Value ένα γεγονός έχει περισσότερες πιθανότητες να είναι τυχαίο. Όσο μικρότερο είναι, τόσο μικρότερες και οι πιθανότητες για την τυχαιότητα του. Σε μία μελέτη ένας ερευνητής μπορεί να ορίσει ο ίδιος το κατώφλι απόφασης για το πότε ένα γεγονός έχει αρκετές πιθανότητες να είναι τυχαίο. Συνήθως αυτό το κατώφλι ορίζεται στο 5% της κανονικής κατανομής. Όπως φαίνεται και στην παρακάτω εικόνα όλα τα P-Values που είναι μικρότερα από μία κρίσιμη τιμή δεν θεωρούνται τυχαία γεγονότα και άρα αποκτούν σημαντικότητα στην ανάλυση.

4.3 Ορισμός και οργάνωση του χώρου μιας hotspot ανάλυσης

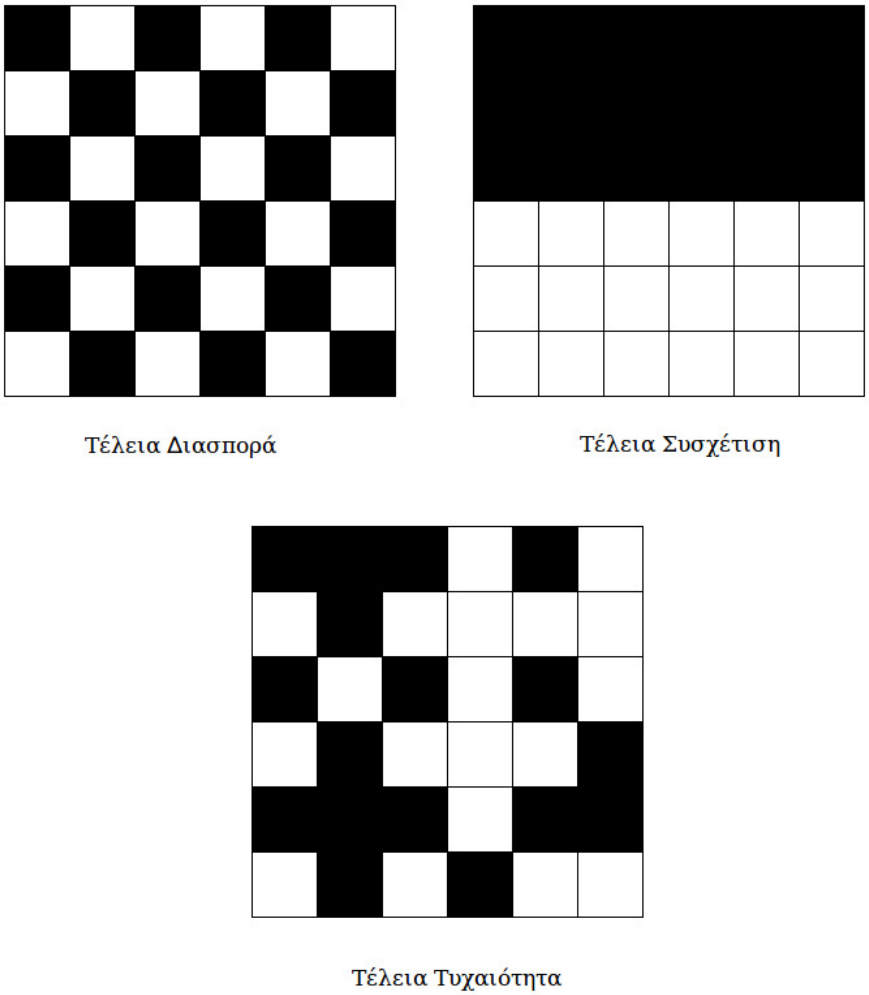
Ο ορισμός και η οργάνωση των ορίων μίας ανάλυσης είναι αρκετά σημαντική διαδικασία. Ένας λάθος ορισμός των ορίων μπορεί να επιφέρει λανθασμένα αποτελέσματα και να καταστήσει την ανάλυση άκυρη. Επίσης αποτελεί γνωστή πρακτική η κατάτμηση του εμβαδού που σχηματίζουν τα όρια της ανάλυσης σε πολύγωνα, άλλες φορές ισόποσα και άλλες φορές διαφορετικά. Για παράδειγμα αν κάποιος θέλει να μελετήσει τα κοινωνικά χαρακτηριστικά μία χώρας θα ήταν δόκιμο να τα χωρίσει κατά περιοχές ή πόλεις ή περιφέρειες και όχι σε ισόποσα τμήματα.

Τα όρια της hotspot analysis ονομάζονται Bounding Box. Κάθε παρατήρηση εκτός αυτού δεν λαμβάνεται υπόψιν. Η κατάτμηση δημιουργεί κελιά (Cells) ενώ πλέον ο χώρος της ανάλυσης μετατρέπεται σε ένα πλέγμα από κελιά (Grid of Cells). Το πόσο μεγάλο ή πόσο μικρό είναι ένα κελί μπορεί να επηρεάσει και αυτό τα αποτελέσματα της ανάλυσης, οπότε και αυτό θα πρέπει να επιλέγεται με προσοχή. Κάθε κελί έχει

την δική του αξία η οποία παράγεται από τις παρατηρήσεις οι οποίες ανήκουν σε αυτό το κελί.

4.4 Moran’s I

Το στατιστικό Moran’s I πρόκειται για μία μετρική χωρικής αυτοσυσχέτισης. Άρα στην ουσία υπολογίζει το πόσο όμοιο είναι ένα σημείο με τα γύρο του. Όπως κάθε μετρική αυτοσυχέτισης παίρνει τιμές από το -1 μέχρι το 1. Όταν το στατιστικό υπολογιστεί -1, τότε σημαίνει ότι τα δεδομένα είναι σε τέλεια διασπορά (Perfect Dispersion), άρα και ομαδοποίηση από ανόμοιες τιμές. Όταν το στατιστικό υπολογιστεί 0, τότε τα δεδομένα είναι σε τέλεια τυχαιότητα (Perfect Randomness), ενώ όταν είναι +1 τα δεδομένα είναι ομαδοποιημένα με όμοιες τιμές (Perfect Correlation). Οι στην εικόνα 4 παρουσιάζονται σχηματικά οι παραπάνω καταστάσεις των δεδομένων κατά το μοντέλο του Moran.



Εικόνα 4: Τέλειες τιμές του Moran's I μοντέλου

Όταν η κατανομή των δεδομένων ακολουθεί την τέλεια συσχέτιση ή την τέλεια διασπορά λέγεται ότι δεν είναι τυχαία κατανεμημένα με αυτόν το τρόπο και ακολουθούν κάποιο μοτίβο.

Σημαντικό ρόλο σε τέτοιου είδους αναλύσεις παίζει η Μηδενική Υπόθεση (Null Hypothesis), η οποία είναι μία φράση που περιγράφει ότι δύο παρατηρήσεις ή δύο ομάδες παρατηρήσεων δεν έχουν καμία συσχέτιση μεταξύ τους. Η ουσία μιας τέτοιας ανάλυσης είναι να καταρρίψει την μηδενική υπόθεση και να βγάλει συμπεράσματα ως προς το αντίθετο. Δηλαδή ότι οι παρατηρήσεις με κάποιον τρόπο συνδέονται δεν είναι τυχαίες και υπάρχει συσχέτιση.

4.5 Getis and Ord G_i^*

Το G_i^* στατιστικό των Getis and Ord είναι ένας τρόπος υπολογισμού Z-Score και έχει ως σκοπό την ανίχνευση μοτίβων σε μία γεωγραφική έκταση. Τα γεγονότα αυτά και πάλι θεωρούνται σημαντικά και μη τυχαία όπως και στο μοντέλο του Moran. Σε αντίθεση με το μοντέλο του Moran που δίνει μία γενική εικόνα για τις παρατηρήσεις σε μία περιοχή μπορεί να μελετήσει και να αποτυπώσει πολύ καλύτερα τοπικά φαινόμενα.

Πριν αναφερθεί ο μαθηματικός τύπος υπολογισμού πρέπει να γίνει κατανοητό ότι για τον υπολογισμό του G_i^* στατιστικού δεν αρκεί μόνο η αξία μίας περιοχής που έχει οριστεί στα πλαίσια της ανάλυσης. Σημαντικό ρόλο παίζουν και οι αξίες γειτονικών περιοχών.

Έστω ότι γίνεται μία μελέτη για το πιο πολυεπισκεπτόμενο σημείο του δήμου Αθηναίων. Και έστω ότι έχουν ομαδοποιηθεί οι παρατηρήσεις της επισκεψιμότητας σε τετράγωνα και έχουν δώσει κάποια αξία σε αυτά. Για τον υπολογισμό του Z-Score ενός τετραγώνου θα ληφθεί υπόψιν και η αξία των γειτονικών τετραγώνων, είτε άμεσων, είτε έμμεσων. Η μέγιστη απόσταση που μπορούν να έχουν οι γείτονες που επηρεάζουν το τελικά αποτέλεσμα ορίζεται από τον ερευνητή.

Έστω ότι υπάρχει μια γεωγραφική έκταση η οποία έχει χωριστεί σε n περιοχές, και μέσα σε αυτή την έκταση υπάρχουν N παρατηρήσεις οι οποίες προσφέρουν αξία X_j στα κελιά που ανήκουν. Και έστω ότι το χωρικό βάρος δύο γειτονικών περιοχών i και j είναι $w_{i,j}$. Το G_i^* στατιστικό δίνεται από την ακόλουθη συνάρτηση:

$$G_i^* = \frac{\sum_{j=1}^{n} w_{i,j}x_j - \overline{X}\sum_{j=1}^{n} w_{i,j}}{S\sqrt{\frac{[n\sum_{j=1}^{n} w_{i,j}^2 - (\sum_{j=1}^{n} w_{i,j})^2]}{n-1}}} \qquad (1)$$

όπου

$$\overline{X} = \frac{\sum_{j=1}^{n} x_j}{n} \qquad (2)$$

και

$$S = \sqrt{\frac{\sum_{j=1}^{n} x_j^2}{n} - (\overline{X})^2} \qquad (3)$$

Κάθε G_i^* που παράγεται είναι ένα Z-Score και άρα δεν χρειάζονται περαιτέρω υπολογισμοί.

4.6 Hotspot analysis στα μεγάλα δεδομένα

Μεγάλος όγκων δεδομένων μπορούν να χρησιμοποιηθούν για hotspot ανάλυση. Μιας και η συγκεκριμένη ανάλυση είναι πλήρως ντετερμινιστική τα μεγάλα δεδομένα βοηθούν στην καλύτερη αποτύπωση των αποτελεσμάτων. Οι γραμμές που ακολουθούν είναι αφιερωμένες σε ορισμούς hotspot analysis πάνω σε διαφορετικούς τύπους μεγάλων δεδομένων.

4.6.1 Hotspot Analysis σε χωρικά δεδομένα

Η hotspot ανάλυση είναι μία μορφή ανάλυσης η οποία βασίστηκε πάνω σε γεωγραφικά δεδομένα για να περιγράψει καταστάσεις και σημαντικά γεγονότα. Στο paper των Getis and Ord περιγράφεται μία μελέτη για ένα φαινόμενο το οποίο συνέβη στην Βόρεια Καρολίνα των Ηνωμένων Πολιτειών της Αμερικής και είχε σχέση με την θνησιμότητα βρεφών μικρότερων από ενός έτους. Η ανάλυση έδειξε ότι υπήρχε μεγάλη θνησιμότητα βρεφών σε μία συγκεκριμένη περιοχή της Βόρειας Καρολίνας και αυτό θα μπορούσε να ερευνηθεί παραπάνω από επιδημιολόγους ή κοινωνιολόγους.

Ακόμα μία μελέτη που παρουσιάζεται στην προαναφερθείσα δημοσίευση είναι η εκτίμηση τιμής των κατοικιών στην πόλη San Diego. Σε αυτή τη περίπτωση ένα προάστιο του San Diego βγήκε εκτός της ανάλυσης καθώς οι τιμές πώλησης κατοικιών ήταν τρεις φορές μεγαλύτερες από αυτές που είχε το αμέσως επόμενο προάστιο. Τελικά τα αποτελέσματα έδειξαν ότι δύο παραθαλάσσιες συνοικίες είχαν

τις κατοικίες με τις υψηλότερες τιμές ενώ στα κεντρικά της πόλης υπήρχαν αρκετές ομαδοποιημένες οι οποίες είχαν τις μικρότερες.

Σε μία άλλη δημοσίευση [22] που παρουσιάζει ενδιαφέρον η ανάλυση έγινε πάνω σε δεδομένα τα οποία είχαν να κάνουν με τροχαία ατυχήματα στα πλαίσια μίας πόλης. Σκοπός της μελέτης ήταν σε πρώτη φάση να αναδειχθούν εκείνοι οι δρόμοι οι οποίοι ήταν πιο επικίνδυνοι για την δημιουργία ενός ατυχήματος. Σε δεύτερη φάση χρησιμοποιώντας τα αποτελέσματα της ανάλυσης ο ερευνητής μπορούσε να κρίνει κατά πόσο επικίνδυνη είναι μία στάση λεωφορείων κοντά σε εκείνα τα σημεία που παρουσίαζαν μεγάλη συγκέντρωση ατυχημάτων, ώστε να προταθεί η μεταφορά της σε ένα άλλο σημείο πιο ασφαλές.

Τέλος στο [14] χρησιμοποιείται hotspot ανάλυση για την μελέτη της χρήσης κινητών τηλεφώνων. Σκοπός είναι η εύρεση περιοχών με μεγάλη πυκνότητα χρήσης όπου στέλνονται μεγάλες ποσότητες δεδομένων. Τα αποτελέσματα την ανάλυσης δείχνουν που υπάρχει τεράστια χρήση των κινητών τηλεφώνων και μπορούν να χρησιμοποιηθούν στην αποτελεσματικότερη λειτουργία του δικτύου μίας εταιρίας παρόχου, ή στην βελτιστοποίηση της στρατηγικής της εισαγωγής νέων κυψελών στο δίκτυο.

4.6.2 Hotspot Analysis σε χώρο-χρονικά δεδομένα

Στα πλαίσια του GISCup 2016 [20] ζητήθηκε μία επέκταση του G_i^* στατιστικού για την hotspot ανάλυση πάνω σε χώρο-χρονικά δεδομένα μέσω της ανάπτυξης ενός αποδοτικού αλγορίθμου. Σκοπός ήταν να βρεθούν σημεία στον χώρο και τον χρόνο που παρουσιάζουν μεγάλη στατιστική σημαντικότητα, δηλαδή έχουν μεγάλο Z-Score και πολύ μικρό P-Value. Η κύρια διαφορά με τον παραδοσιακό τρόπο του hotspot analysis ήταν η εισαγωγή του χρόνου στα δεδομένα που χρησιμοποιήθηκαν.

Για να επιτευχθεί η ανάλυση στους τρεις πλέον άξονες του προβλήματος (μήκος, πλάτος, χρόνος) τα όρια κατασκευάστηκαν ως ένας κύβος του οποίου το μήκος και το πλάτος συμβόλιζαν τα γεωγραφικά μήκη και πλάτη και το ύψος του τον χρόνο. Τα αποτελέσματα της ανάλυση έδειξαν ότι μπορούν να βγουν σημαντικά ευρήματα για φαινόμενα τα οποία παρατηρούνται σε χώρο-χρονικά δεδομένα.

Σημαντικά ευρήματα παρουσιάζονται και στο [13] το οποίο ασχολείται με την δημιουργία Black Holes (Hot Spots) και Volcanos σε μία γεωγραφική έκταση για να αναγνωριστούν ανωμαλίες στα μοτίβα που έχουν οι άνθρωποι στις καθημερινές τους μετακινήσεις. Πιο συγκεκριμένα αναφέρει ότι τα Black Holes είναι γεωγραφικά τμήματα τα οποία παρουσιάζουν μεγάλο αριθμό κίνησης παρατηρήσεων προς αυτά συναρτήσει του χρόνου, δηλαδή το συνολικό πλήθος των παρατηρήσεων που εισέρχεται στην περιοχή είναι μεγαλύτερο από τον συνολικό αριθμό που εξέρχεται. Ενώ τα Volcanos ορίζονται ως γεωγραφικά τμήματα τα οποία παρουσιάζουν μείωση αυτής της συγκέντρωσης. Επίσης αναφέρει ότι κάθε σημείο το οποίο είναι Black Hole κάποια στιγμή στο μέλλον θα μετατραπεί σε Volcano του οποίου οι εκροές θα δημιουργήσουν νέα Black Holes κάπου αλλού.

4.6.2 Hotspot Analysis Τροχιών

Η ανάλυση δεδομένων που περιέχουν τροχιές μπορεί να γίνει αρκετά απαιτητική. Ποιος είναι ο βέλτιστος τρόπος κατανομής των παρατηρήσεων σε ένα grid? Με ποιόν τρόπο θα κρατηθεί περισσότερη πληροφορία από την κίνηση των αντικειμένων? Όλα αυτά είναι ερωτήματα τα οποία δημιουργούνται όταν κάποιος θέλει να επεξεργαστεί δεδομένα τροχιών. Οι προαναφερθέντες τρόποι ανάλυσης δεν θα μπορούσαν να πετύχουν σωστά αποτελέσματα καθώς επεξεργάζονται μεμονωμένα τα σημεία των παρατηρήσεων και όχι ως ένα σύνολο από σημεία τα οποία σχηματίζουν μία τροχιά.

Στις επόμενες παραγράφους θα περιγραφούν τρόποι που υλοποιήθηκαν ώστε να λαμβάνουν υπόψιν τα σημεία ως σύνολο και να αντλούν πληροφορία από την τροχιά που σχηματίζει ένα κινητό αντικείμενο μέσα σε ένα grid, επεκτείνοντας την λειτουργικότητα του BigCAB αλγορίθμου.

5. **Hot Spot Analysis σε Spark**

Αυτό το κεφάλαιο είναι αφιερωμένο στην δομή του προβλήματος της hotspot ανάλυσης χώρο-χρονικών δεδομένων και τροχιών, και στην πρόταση τρόπων επίλυσης τους. Επίσης περιγράφεται η δομή του συνόλου των δεδομένων που αναλύθηκε, ο τρόπος δημιουργίας του χώρου της ανάλυσης και παρουσιάζονται οι αλγόριθμοι που χρησιμοποιήθηκαν σε μορφή ψευδοκώδικα.

5.1 Hotspot analysis χώρο-χρονικών δεδομένων σε Spark

5.1.1 Αποτύπωση του προβλήματος

Έστω ένα dataset D το οποίο αποτελείται από χώρο-χρονικά σημεία/παρατηρήσεις p τα οποία εκφράζονται από τις χωρικές συντεταγμένες p.x και p.y, την χρονική σήμανση p.t και μία τιμή p.v η οποία αναπαριστά την αξία της παρατήρησης p. Επίσης, έστω ότι έχουν οριστεί τα χώρο-χρονικά όρια της ανάλυσης B και η κατάτμηση αυτού στον χώρο και τον χρόνο σε P κατατμήσεις, οι οποίες δημιουργούν n κελιά c. Υπάρχει μία προς μίας συσχέτιση μεταξύ μίας παρατήρησης p με ένα κελί c_i το οποίο αποφασίζεται από το αν η παρατήρηση p εμπεριέχεται στο κελί c_i. Επίσης ορίζεται η τιμή x_i ως το άθροισμα των p.v που εμπεριέχονται σε ένα κελί c_i .

Το πρόβλημα αναφέρεται στον εντοπισμό των στατιστικά πιο σημαντικών χώρο-χρονικών περιοχών c_i, όπου η σημαντικότητα ορίζεται ως μία συνάρτηση του x_i ενός κελιού c_i αλλά και των αντίστοιχων τιμών που έχουν τα γειτονικά κελιά του c_i. Ο τύπος υπολογισμού φαίνεται στην συνάρτηση 1 και είναι το στατιστικό G_i^*. Επίσης για τον υπολογισμό του στατιστικού θα γίνουν κάποιες υποθέσεις για την απλούστευση του προβλήματος. Για τον υπολογισμό του G_i^* τα μόνα γειτονικά κελιά $N(c_i)$ που θα χρησιμοποιηθούν θα είναι μόνο τα άμεσα γειτονικά. Επίσης το βάρος $w_{i,j}$ των δύο γειτονικών κελιών c_i και c_j θεωρείται ότι θα είναι ίσο με την μονάδα. Οπότε η συνάρτηση 1 μπορεί να απλοποιηθεί στην συνάρτηση 4 που φαίνεται παρακάτω.

$$G_i^* = \frac{\sum_{j \in \mathcal{N}(c_i)} x_j - \overline{X} \cdot |\mathcal{N}(c_i)|}{S \cdot \sqrt{\frac{n \cdot |\mathcal{N}(c_i)| - |\mathcal{N}(c_i)|^2}{n-1}}} \qquad (4)$$

5.1.2 Μορφή και Δομή Δεδομένων

Τα δεδομένα που δέχεται ως ορίσματα ο BigCAB αλγόριθμος πρόκειται για CSV αρχεία τα οποία περιέχουν πληροφορία για κούρσες από τα κίτρινα ταξί της Νέας Υόρκης. Αυτά τα αρχεία μπορούν να βρεθούν στο [21]. Το τελικό σύνολο των δεδομένων που χρησιμοποιήθηκε αναφερόταν στα στοιχεία της χρονιάς του 2015.

Τα αρχεία περιέχουν δεδομένα που περιγράφουν κάθε κούρσα ενός ταξί ξεχωριστά αλλά και χαρακτηριστικά για την ίδια την διαδρομή. Το δεδομένα που είχαν αξία τελικά για τον αλγόριθμο και το πρόβλημα που δόθηκε είναι τα ακόλουθα:

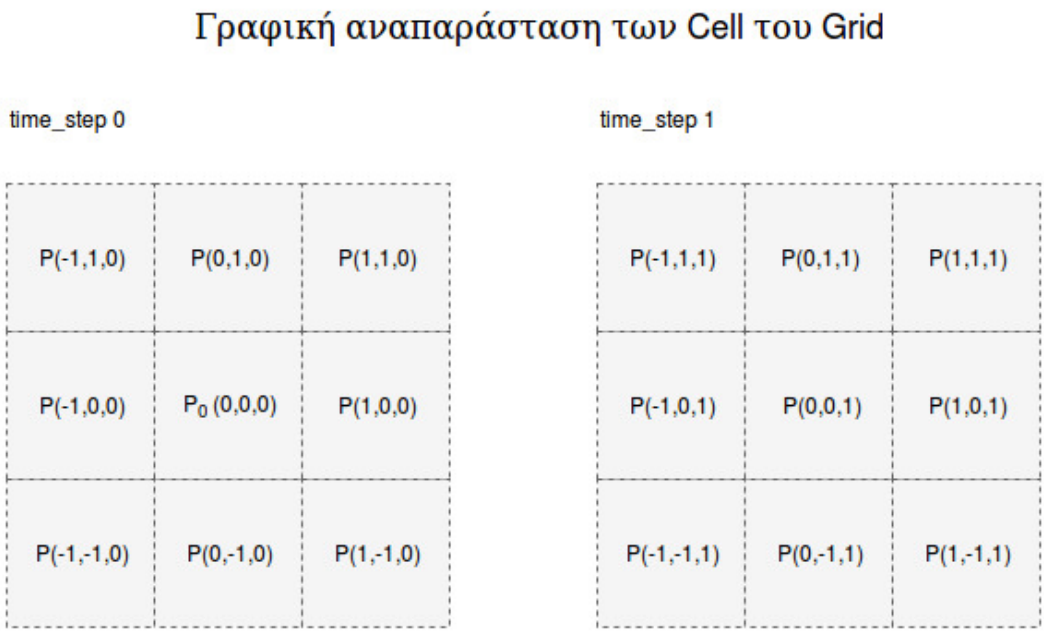
- tpep_dropoff_datetime: η χρονική στιγμή του τέλους της κούρσας
- Passenger_count: ο αριθμός των επιβαινόντων/πελατών της κάθε κούρσας
- Dropoff_longitude: η τεταγμένη του σημείου προορισμού
- Dropoff_ latitude: η τετμημένη του σημείου προορισμού

5.1.3 Όρια ανάλυσης και δημιουργία του χώρο-χρονικού κύβου

Τα χωρικά όρια της ανάλυσης ήταν ένα νοητό παραλληλόγραμμο το οποίο περικλείει την πόλη της Νέας Υόρκης. Τα χρονικά όρια βασίστηκαν στα χρονικά όρια του συνόλου των δεδομένων, δηλαδή 1/1/2015:00:00 με 1/1/2016:00:00. Ο συνδυασμός αυτών των δύο δημιουργεί τον χώρο-χρονικό κύβο.

Επίσης ορίστηκε το σημείο εκκίνησης υπολογισμού των κελιών που θα δημιουργούνταν, στο πεδίο του χώρου ως το σημείο με συντεταγμένες (0,0) και στο πεδίο του χρόνου 1/1/2015:00:00. Άρα αν αυτό το σημείο ήταν εντός των πλαισίων της ανάλυσης θα ήταν το κελί με x,y,t (0,0,0). Η λογική της κατάτμησης των κελιών και της αντιστοίχησης τους με τις τριπλέτες (x,y,t) παρουσιάζεται στην παρακάτω.

Έστω ότι έχει οριστεί ένα σημείο εκκίνησης του σχηματισμού του Grid $P_0(x_0,y_0,t_0)$ και τα μεγέθη των κελιών, στον χώρο cell_size ∂d (αξίζει να σημειωθεί ότι το ∂d είναι ίσο για την τεταγμένη και την τετμημένη) και στον χρόνο time_step ∂t . Και τα όρια την ανάλυσης $P_{min}(x_{min},y_{min},t_{min})$ και $P_{max}(x_{max},y_{max},t_{max})$. Τότε το Grid που θα σχηματιστεί θα έχει ως σημείο εκκίνησης το συγκεκριμένο κελί $c_0(x_0,y_0,t_0)$ το οποίο θα έχει μέγεθος $\partial d \times \partial d \times \partial t$ και τα υπόλοιπα κελιά θα υπολογιστούν βάσει αυτής της αρχής έχοντας τις ίδιες διαστάσεις μέχρι τα όρια τις ανάλυσης P_{min} και P_{max}. Η γραφική αναπαράσταση των κελιών παρουσιάζεται στην παρακάτω εικόνα.



Εικόνα 5: Σχηματισμός Κελιών στο Grid

Άρα τελικά θα δημιουργηθεί ένας κύβος ο οποίος θα έχει $x_{segments}=\frac{P_{max}.x-P_{min}.x}{\partial d}$ κατατμήσεις στον άξονα των x, $y_{segments}=\frac{P_{max}.y-P_{min}.y}{\partial d}$ κατατμήσεις στον άξονα των y και $t_{segments}=\frac{P_{max}.t-P_{min}.t}{\partial t}$ κατατμήσεις στον άξονα των τ. Τότε ο συνολικός αριθμός των κελιών που θα έχουν δημιουργηθεί θα υπολογίζεται από τον τύπο:

$$N_{cells}=x_{segments}\times y_{segments}\times t_{segments}$$

Τέλος για την ευκολία των υπολογισμών του αλγορίθμου αλλά και της αύξησης της αποδοτικότητας σε κάθε κελί $c_i(x_i,y_i,t_i)$ μπορεί να αποδοθεί ένα μοναδικό κλειδί το οποίο υπολογίζεται από τον τύπο:

$$Id_{cell}=x_i+y_i\times x_{segments}+t_i\times x_{segments}\times y_{segments}$$

5.1.4 Αλγόριθμος BigCAB

Ο αλγόριθμος BigCAB είχε σαν σκοπό την ανάλυση χώρο-χρονικών δεδομένων με αποδοτικό τρόπο. Στο κεφάλαιο αυτό περιγράφεται η λειτουργία του BigCAB μιας και

πάνω στην επέκταση αυτού του αλγορίθμου βασίστηκε η ανάλυση των δεδομένων με τροχιές.

Αρχικά θα πρέπει να αναφερθεί ότι ο BigCAB εκτελεί Hotspot Analysis πάνω στα δεδομένα με σκοπό να βρει τα top-k κελιά με την υψηλότερη Z-Score τιμή. Δηλαδή όπως προαναφέρθηκε, κατά την παρουσίαση του hotspot analysis σε προηγούμενο κεφάλαιο, τα κελιά που παρουσιάζουν μεγάλη στατιστική σημαντικότητα λόγω της μεγάλης απομάκρυνσης της τιμής τους από τον μέσο όρο και την εξαιρετικά χαμηλή τυχαιότητα τους. Άλλη μία παρατήρηση είναι ότι το συγκεκριμένο Hotspot Analysis γίνεται με μία απλοποιημένη μορφή του Getis-Ord G_i^* στατιστικού το οποίο δημιουργείτε αν τεθούν όλα τα βάρη των γειτόνων κελιών ($w_{i,j}$) ίσα με μονάδα και προσμετρηθούν μόνο η άμεσοι γείτονες κάθε κελιού.

Το dataset που είχε να αναλύσει ο αλγόριθμος ήταν μία σειρά από CSV αρχεία τα οποία περιείχαν δεδομένα από κούρσες ταξί της Νέας Υόρκης για ένα συγκεκριμένο χρονικό διάστημα. Τα δεδομένα που χρησιμοποιήθηκαν ήταν η τεταγμένη, η τετμημένη του τέλους της κούρσας των ταξί καθώς και την χρονική στιγμή την οποία παρατηρήθηκε αυτό το φαινόμενο και τον αριθμό των ατόμων τα οποία επέβαιναν το ταξί κατά την διάρκεια της κούρσας. Τα όρια της ανάλυσης ήταν ορισμένα εκ των προτέρων και ήταν ίσα με τα όρια της πόλης της Νέας Υόρκης οπότε ο αλγόριθμος θα έπρεπε να μην λάβει υπόψιν οποιοδήποτε δεδομένο εκτός των συγκεκριμένων ορίων. Το μέγεθος του κάθε κελιού μπορούσε να δοθεί από τον χρήστη ως ορίσματα του αλγορίθμου είτε επηρεάζοντας το μέγεθος του της χωρικής συνισταμένης, είτε της χρονικής. Κάθε κελί συμβολίζει μια γεωγραφική περιοχή μέσα σε ένα διάστημα χρόνου.

Ο αλγόριθμος του BigCAB μπορεί να χωριστεί σε 2 MapReduce φάσεις τις οποίες ακολουθεί μία for-each φράση (clause) για την εκτίμηση του Z-Score και μία πράξη κατάταξης (Sorting) για να βρεθούν τα 50 στατιστικά πιο σημαντικά cells. Επίσης μεταξύ των MapReduce υπάρχει ακόμη μία for-each φράση υπεύθυνη για τον υπολογισμό απαραίτητων στατιστικών για την εκτίμηση των G_i^* στατιστικών. Στην παρακάτω εικόνα παρουσιάζεται η υλοποίηση του BigCAB σε μορφή ψευδοκώδικα.

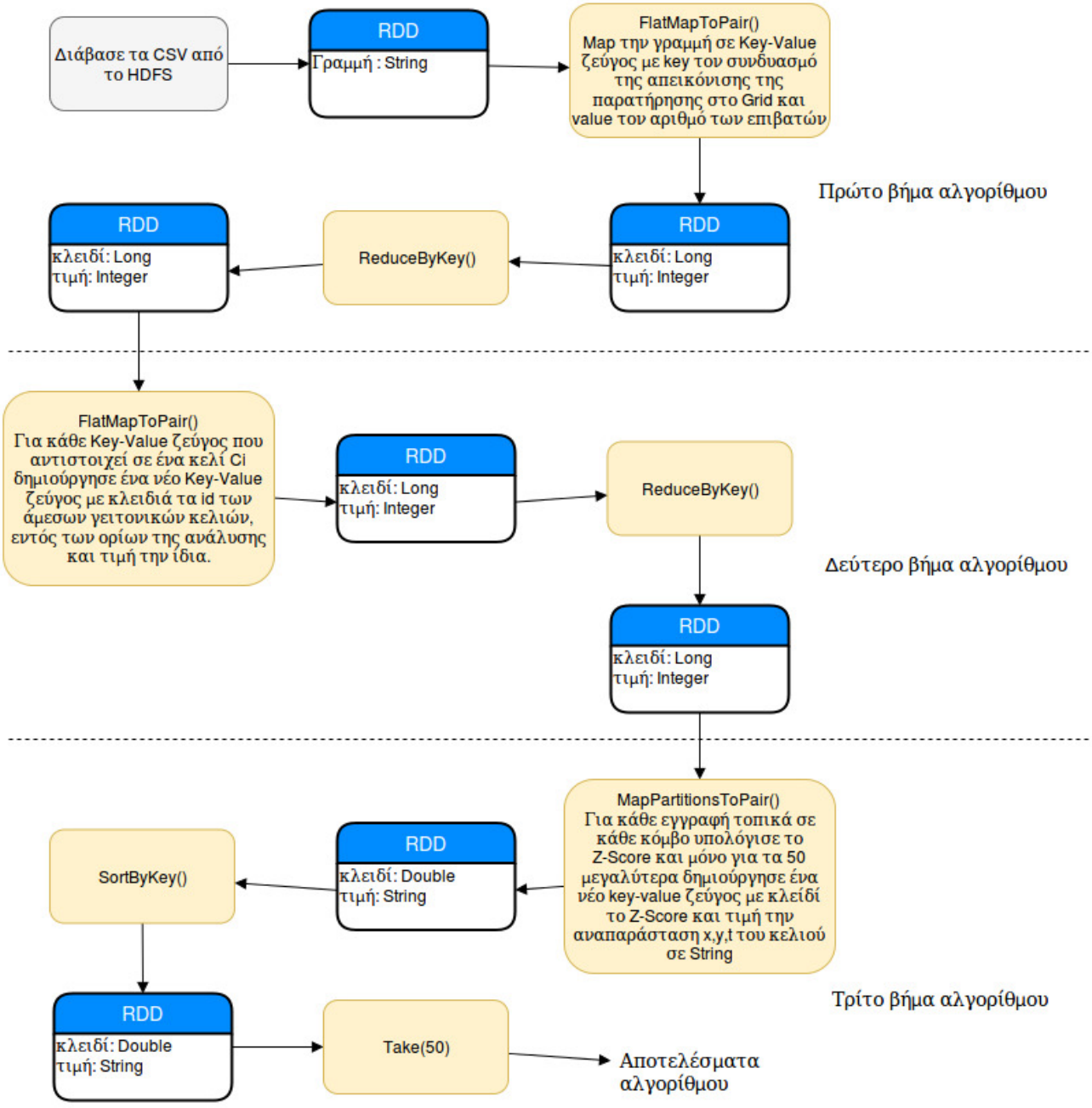
Algorithm 1 BigCAB algorithm

```
1: procedure BIGCAB
2: gridRDD = filesOnHDFS.flatMapToPair(p =>
3:     emit new pair(getCellId(p), p.v)
4: ).reduceByKey(p1, p2 => emit p1.v + p2.v)
5: gridRDD.forEach(ci => update accumulators)
6: neighborRDD = gridRDD.flatMapToPair(ci =>
7:     N(ci) = getDirectNeighborIds(ci)
8:     for each j in N(ci) do
9:         emit new pair(j, xi)
10:    end for
11: ).reduceByKey(x1, x2 => emit x1 + x2)
12: scoresRDD=neighborRDD.mapPartitionsToPair(ci=>
13:    for each local ci in neighborRDD do
14:        list.add(new pair(ci, Gi*))
15:        sort list and keep only the top-k pairs
16:    end for
17:    emit list)
18: sort scoresRDD and return the top-k cells
19: end procedure
```

Αλγόριθμος 1: BigCAB Αλγόριθμος

Το πρώτο MapReduce έχει σαν στόχο την ανάγνωση των αρχείων από το HDFS, την μετατροπή αυτών σε ένα ζεύγος κλειδιού-τιμής (Map φάση) και την καταμέτρηση των τιμών από τα ζεύγη με τα ίδια κλειδιά (Reduce φάση). Πιο αναλυτικά το Map γίνεται διαβάζοντας από τα CSV αρχεία γραμμή, γραμμή, την τεταγμένη, την τετμημένη και την χρονική άφιξης στον προορισμό και τον αριθμό των επιβαινόντων. Τα τρία πρώτα ορίσματα κάθε γραμμής χρησιμοποιούνται για να κατασκευάσει ένα κλειδί τύπου Long το οποίο θεωρείται και το μοναδικό αναγνωριστικό κάθε κελιού του grid. Ο αριθμός των επιβαινόντων κάθε κούρσας χρησιμοποιείται ως τιμή τους ζεύγους κλειδιού-τιμής. Στην Reduce φάση αθροίζονται όλες οι τιμές των ζευγών με τα ίδια κλειδιά για να σχηματίσουν την αξία που έχει κάθε κελί από μόνο του. Υπενθυμίζεται ότι στον υπολογισμό του G_i^* στατιστικού παίζουν σημαντικό ρόλο και αξίες των γειτονικών κελιών.

Στην συνέχεια υπολογίζονται το άθροισμα και το άθροισμα των τετραγώνων όλων των τιμών του grid. Αυτό συμβαίνει για να υπολογιστούν τα στατιστικά που χρειάζονται για την εκτίμηση του G_i^*. Αυτά είναι μέση τιμή και η διακύμανση από την μέση τιμή.



Εικόνα 6: Διάγραμμα Ροής BigCAB

Η επόμενη MapReduce εργασία είναι πολύ σημαντική για τον υπολογισμό του Z-Score. Είναι το βήμα στο οποίο κάθε κελί μεταδίδει την αξία του στα άμεσα γειτονικά του και επιτυγχάνεται δημιουργώντας $N(c_i)$ νέα ζεύγη κλειδιού-τιμής με

κλειδί το μοναδικό αναγνωριστικό των γειτονικών κελιών και τιμή την αξία που έχει το κεντρικό κελί. Άρα κάθε ζεύγος κλειδιού-τιμής θα δημιουργήσει το πολύ 26 νέα ζεύγη τα οποία στην συνέχεια θα αθροιστούν από την Reduce φάση. Στο τέλος αυτής της διαδικασίας η τιμή του κάθε ζεύγους θα έχει αξία ίση με το άθροισμα των των X_j και ο αλγόριθμος θα είναι έτοιμος να υπολογίσει το Z-Score.

Στο επόμενο βήμα υπολογίζεται τελικά το Z-Score κάθε κελιού τοπικά σε κάθε κόμβο του cluster ενώ ταυτόχρονα κρατούνται μόνο τα 50 με τα μεγαλύτερα G_i^* του κάθε κόμβου. Τελικά μόνο αυτά επιστρέφονται και κατατάσσονται σε φθίνουσα σειρά για να βρεθούν τα 50 στατιστικά πιο σημαντικά κελιού του grid στον χώρο-χρόνο. Στη εικόνα Χ φαίνεται το Διάγραμμα Ροής του αλγορίθμου για την διευκόλυνση του αναγνώστη

5.2 Hotspot Analysis τροχιών σε Spark

5.2.1 Αποτύπωση του προβλήματος

Έστω ένα dataset D το οποίο αποτελείται από τροχιές T οι οποίες εκφράζονται από ένα σύνολο από χώρο-χρονικά σημεία/στιγμιότυπα p_i, όπου $\{p_1, ... p_k\}$ ανήκει στο T. Επίσης έστω ότι το χώρο-χρονικό στιγμιότυπο της τροχιάς pi περιέχει τις χωρικές συντεταγμένες $p_i.x$ και $p_i.y$ και την χρονική σήμανση $p_i.t$ της παρατήρησης. Επιπροσθέτως, έστω ότι έχουν οριστεί τα χώρο-χρονικά όρια της ανάλυσης B και η κατάτμηση αυτού στον χώρο και τον χρόνο σε P κατατμήσεις, οι οποίες δημιουργούν n κελιά c. Υπάρχει μία προς πολλά συσχέτιση μεταξύ μίας τροχιάς T με τα κελιά c αλλά μόνο μία προς μία συσχέτιση ενός στιγμιότυπου p_i που ανηκει στην T με ένα κελί c_i. Επίσης ορίζεται η τιμή x_i ως ένα άθροισμα των αξιών που έχει να προσδώσει κάθε τροχιά T η οποία περιέχει έστω ένα στιγμιότυπο p_i που εμπεριέχεται σε ένα κελί c_i.

Το πρόβλημα αναφέρεται στον εντοπισμό των στατιστικά πιο σημαντικών χώρο-χρονικών περιοχών c_i, όπου η σημαντικότητα ορίζεται ως μία συνάρτηση του x_i ενός κελιού c_i αλλά και των αντίστοιχων τιμών που έχουν τα γειτονικά κελιά του c_i. Ο τύπος υπολογισμού φαίνεται στην συνάρτηση 1 και είναι το στατιστικό G_i^*. Επίσης για τον υπολογισμό του στατιστικού χρησιμοποιήθηκε η απλοποιημένη μορφή που φαίνεται στην συνάρτηση 4 αλλά και η συνάρτηση 1.

5.2.2 Δομή δεδομένων

Τα δεδομένα πρόκειται για CSV αρχεία τα οποία περιγράφουν την κίνηση πλοίων στο επίπεδο και στον χρόνο. Η συνιστώσα του χώρου ή επιπέδου περιγράφεται από τις συντεταγμένες των πλοίων στο παγκόσμιο σύστημα συντεταγμένων [WGS84] το

οποίο χρησιμοποιείται από τα περισσότερα μέσα πλοήγησης. Ο χρόνος είναι σε μορφή χρονικής σήμανσης και εκφράζεται σε nanoseconds. Επίσης περιέχουν ένα μοναδικό αναγνωριστικό το οποίο χρησιμοποιείται για τον εντοπισμό των επιμέρους τροχιών μέσα στο σύνολο των δεδομένων. Όλα αυτά τα χαρακτηριστικά χρησιμοποιήθηκαν από τον αλγόριθμο για τον υπολογισμό του G_i^* στατιστικού.

Ο αριθμός των αρχείων είναι 10 όπου το καθένα έχει περίπου 9 GB πληροφορία και έχει τεμαχιστεί σε blocks των 128 MB μέσα στους κόμβους του HDFS. Επίσης ο συνολικός αριθμός των εγγραφών μέσα στα αρχεία είναι 1.934 δισεκατομμύρια.

5.2.3 Όρια ανάλυσης και δημιουργία του χώρο-χρονικού κύβου

Για την ανάλυση των δεδομένων ορίστηκαν δύο διαφορετικά όρια. Το πρώτο έγινε για να αναλυθούν όλα τα δεδομένα που εμπεριέχονταν στα αρχεία, άρα τα όρια της ανάλυσης καθορίζονταν από τις ελάχιστες και μέγιστες τιμές των τροχιών των πλοίων στο πεδίο του χώρου και του χρόνου και υπολογίζονταν σε ένα βήμα προ-επεξεργασίας των δεδομένων κατά την διάρκεια της εκτέλεσης του αλγορίθμου.

Ο δεύτερος ορισμός των ορίων της ανάλυσης έγινε για λόγους μεγάλης συσσώρευσης παρατηρήσεων στο λιμάνι του Πειραιά, το οποίο μονοπωλούσε τα top-k κελιά, και επηρέαζε σε μεγάλο βαθμό τα αποτελέσματα της ανάλυσης. Τα όρια έτσι ορίστηκαν ως ένα νοητό παραλληλόγραμμο το οποίο περικλείει τα νησιά του Αιγαίου. Τέλος τα όρια στον χρόνο υπολογίζονται ως ένα βήμα προ-επεξεργασίας των δεδομένων και στις δύο περιπτώσεις και είναι ίσα με την ελάχιστη και την μέγιστη παρατήρηση μίας τροχιάς στο σύνολο των δεδομένων που διαβάζονται. Ενδεικτικά τα χρονικά όρια της ανάλυσης είναι από τις 29 Μαρτίου του 2007 μέχρι τις 31 Μαρτίου του 2010

Η διαδικασία της κατάτμησης του χώρου της ανάλυσης σε κελιά για να σχηματιστεί το πλέγμα παρέμεινε η ίδια.

5.2.4 Επέκταση του BigCAB Αλγορίθμου

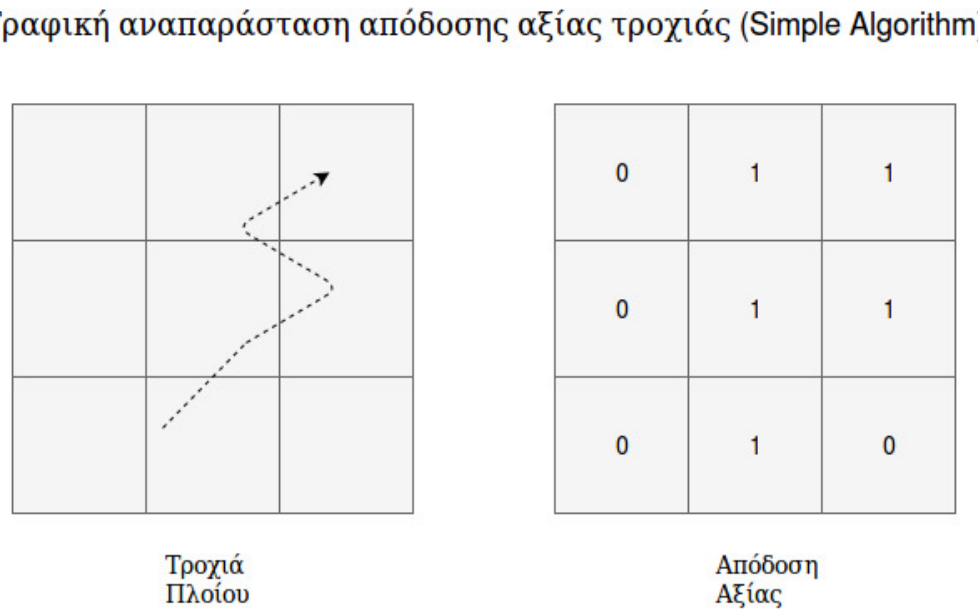
Ως είχε ο αλγόριθμος BigCAB δεν θα ήταν ικανός να πραγματοποιήσει αξιόπιστη ανάλυση πάνω σε τέτοιας μορφής δεδομένα. Υπενθυμίζεται ότι ο αλγόριθμος κατασκευάστηκε για χώρο-χρονικά δεδομένα τα οποία δεν εμπεριείχαν την έννοια της τροχιάς. Γι’αυτό τον λόγο αναδιαμορφώθηκε και επεκτάθηκε. Στα επόμενα υποκεφάλαια θα παρουσιαστούν οι επεκτάσεις του αλγορίθμου με την τελευταία να είναι και η κυριότερη. Αξίζει να σημειωθεί ότι τα όρια της ανάλυσης και στους τρεις ακόλουθους αλγορίθμους υπολογίζονται κατά την διάρκεια του τρεξίματος σε ένα βήμα προ-επεξεργασίας αλλά μπορούν να παραλλαχθούν και από τον προγραμματιστή. Και σε αυτούς τους αλγορίθμους ο χρήστης μπορεί να αλλάξει και

να επηρεάσει το μέγεθος των κελιών που δημιουργούνται με τον ίδιο ακριβώς τρόπο όπως στον BigCAB. Επίσης κάθε κελί συμβολίζει μια γεωγραφική περιοχή μέσα σε ένα διάστημα χρόνου όπως στον BigCAB.

5.2.5 Simple-Mind Algorithm

Ο πρώτος αλγόριθμος είναι και ο πιο απλός. Μερικές μόνο μετατροπές έγιναν στον αλγόριθμο του BigCAB. Η ανάλυση που παρείχε πάνω στα δεδομένα ήταν αρκετά αφελής.

Η αλλαγή που έγινε αναφέρεται στην κατάτμηση των τροχιών σε χώρο-χρονικά κομμάτια. Αν υπάρχει έστω και μία παρατήρηση της τροχιάς ενός πλοίου μέσα σε ένα κελί, αυτό δρα προσθετικά στην αξία του κελιού κατά μία μονάδα. Αλλά αυτό θα μπορούσε να συμβεί μόνο μία φορά καθώς άλλες παρατηρήσεις του ίδιου πλοίου θα απορρίπτονταν. Άρα το πρώτο σημαντικό πρόβλημα είναι πως θα βρεθεί ένας αποδοτικός τρόπος να αναγνωριστούν οι παρατηρήσεις πλοίων που έχουν ήδη προσμετρηθεί μέσα σε ένα κελί. Το συγκεκριμένο αρχεία προσφέρει και το μοναδικό αναγνωριστικό κάθε πλοίου. Οπότε χρησιμοποιώντας αυτό στον σχηματισμό των ζευγών κλειδιών-τιμών θα γινόταν να αναγνωριστεί εάν κάποιο πλοίο έχει προσμετρηθεί ήδη στην αξία ενός κελιού. Η απόδοση αξίας της τροχιάς ενός πλοίου στα κελιά του grid φαίνεται στην παρακάτω εικόνα.



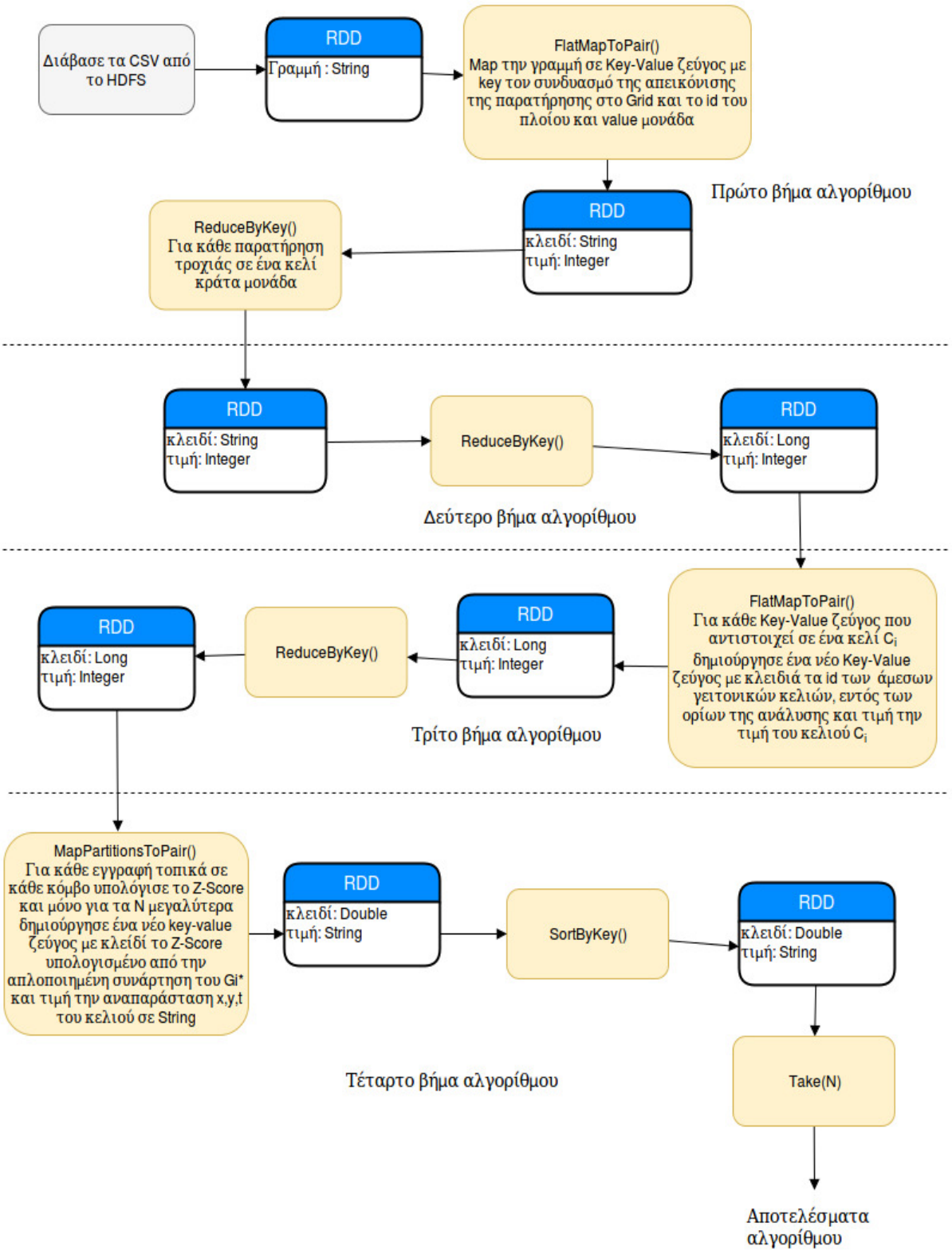
*Εικόνα 7: Αναπαράσταση απόδοσης αξίας τροχιάς στα κελιά που διανύει (*Simple-Mind Algorithm*)*

Πιο συγκεκριμένα η υλοποίηση του πρώτου αλγορίθμου εισάγει ένα επιπλέον βήμα MapReduce πριν το πρώτο MapReduce του BigCAB το οποίο έχει σαν σκοπό να μετασχηματίσει τα δεδομένα που διαβάζονται σε ζευγάρια κλειδιού-τιμής με κλειδί τον συνδυασμό του μοναδικού αναγνωριστικού του κελιού που έγινε η παρατήρηση με το μοναδικό αναγνωριστικό του πλοίου και τιμή την μονάδα. Η φάση του Reduce που έπεται δεν υπάρχει για να αθροίσει όλες τις μονάδες των ζευγών με τα ίδια κλειδιά αλλά για να τις κάνει μοναδικές. Αφού ολοκληρωθεί αυτό το βήμα μπορεί να γίνει ο μετασχηματισμός των ζευγών κλειδιών-τιμών στην μορφή της ανάλυσης του BigCAB ώστε να συνεχιστεί η ανάλυση ως έχει. Αυτό επιτυγχάνεται με την αφαίρεση του μέρους του κλειδιού που δείχνει προς το μοναδικό αναγνωριστικό του πλοίου.

Algorithm 2 Simple-Mind algorithm

```
1: procedure SIMPLEMIND
2: trajectoryRDD = filesOnHDFS.flatMapToPair(p =>
3:     emit new pair((getCellId(p)-p.id).toString(), 1)
4: ).reduceByKey(p1, p2 => emit 1)
5: gridRDD = trajectoryRDD.mapToPair(tr =>
6:     emit new pair ((getCellId(tr.id), 1)
7: ).reduceByKey(tr1, tr2 => emit tr1.v + tr2.v)
8: gridRDD.forEach(ci => update accumulators)
9: neighborRDD = gridRDD.flatMapToPair(ci =>
10:    N(ci) = getDirectNeighborIds(ci)
11:    for each j in N(ci) do
12:        emit new pair(j, xi)
13:    end for
14: ).reduceByKey(x1, x2 => emit x1 + x2)
15: scoresRDD=neighborRDD.mapPartitionsToPair(ci=>
16:    for each local ci in neighborRDD do
17:        list.add(new pair(ci, G*i))
18:        sort list and keep only the top-k pairs
19:    end for
20:    emit list)
21: sort scoresRDD and return the top-k cells
22: end procedure
```

Αλγόριθμος 2: Simple-Mind Αλγόριθμος



Εικόνα 8: Διάγραμμα Ροής Simple-Mind *Algorithm*

5.2.6 Trajectory-aware Algorithm

Η δεύτερη υλοποίηση / μετατροπή του BigCAB αλλάζει ριζικά τον αλγόριθμο και θέτει την έννοια των τροχιών μέσα στην ανάλυση. Όπως ο πρώτος αλγόριθμος έτσι και αυτός μετασχηματίζει τα δεδομένα που διαβάζει σε ζεύγη κλειδιών-τιμών με το κλειδί να παραμένει ίδιο ωστόσο προσθέτει στην τιμή ένα αντικείμενο του οποίου ρόλος είναι να κρατάει την μικρότερη και την μεγαλύτερη χρονική παρατήρηση που έχει συναντήσει σε κάθε χώρο-χρονικό κελί. Σε πρώτη φάση η μέγιστη και η ελάχιστη τιμή είναι ίσες. Στην συνέχεια είναι η σειρά του Reduce να βρει τις ελάχιστες και μέγιστες τιμές όλων των παρατηρήσεων μίας τροχιάς σε ένα κελί.

Το επόμενο βήμα είναι το δεύτερο MapReduce το οποίο στόχο έχει τον μετασχηματισμό του ζεύγους κλειδιού τιμής του πρώτου βήματος στην μορφή που πρέπει να έχει για να γίνει η ανάλυση. Αυτό επιτυγχάνεται αφαιρώντας το κομμάτι του μοναδικού αναγνωριστικού του πλοίου από το κλειδί (όπως γίνεται στον προηγούμενο αλγόριθμο) και την μετατροπή του αντικειμένου με την ελάχιστη και την μέγιστη τιμή των παρατηρήσεων που αθροίστηκαν σε έναν double. Αυτό γίνεται με τον ακόλουθο τύπο.

$$\frac{timestamp_{max} - timestamp_{min}}{c_i.t_{max} - c_i.t_{min}}$$

όπου $timestamp_{max}$ και $timestamp_{min}$ είναι η μέγιστη και η ελάχιστη χρονική παρατήρηση μια τροχιάς στα πλαίσια ενός κελιού και είναι πάντα μικρότερο ή ίσο με ∂t και $c_i.t_{max}$ και $c_i.t_{min}$ τα χρονικά όρια ενός κελιού c_i και είναι ίσο με ∂t .

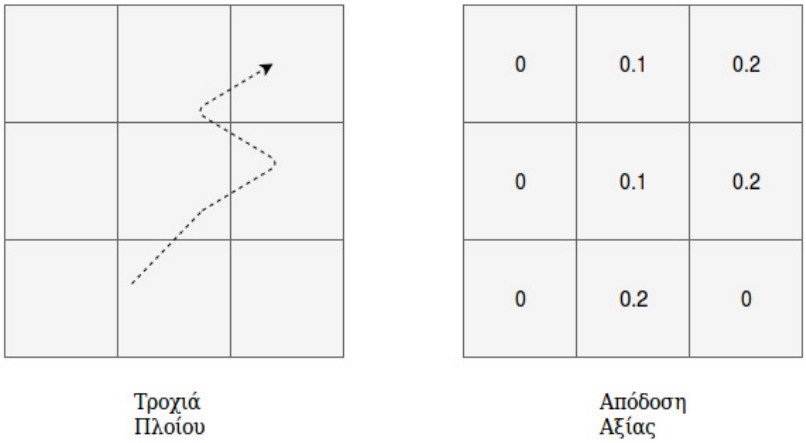
Άρα η συνολική αξία ενός κελιού x_i υπολογίζεται από τον τύπο:

$$x_i = \sum_{j=0}^{n} \frac{timestamp_{max} - timestamp_{min}}{c_i.t_{max} - c_i.t_{min}}$$

όπου το n είναι ο συνολικός αριθμός των τροχιών που περνάνε από ένα κελί c_i και δρουν προσθετικά στην αξία του.

Στην εικόνα 9 φαίνεται ένα παράδειγμα απόδοσης της αξίας μιας τροχιάς στα κελιά στα πλαίσια ενός ∂t . Επίσης αξίζει να σημειωθεί ότι το άθροισμα όλων των τιμών που μπορεί να αποδώσει μία τροχιά στα κελιά που διασχίζει στα πλαίσια ενός ∂t δεν μπορεί να ξεπερνάει την μονάδα.

Γραφική αναπαράσταση απόδοσης αξίας τροχιάς (Trajectory-Aware Algorithm)



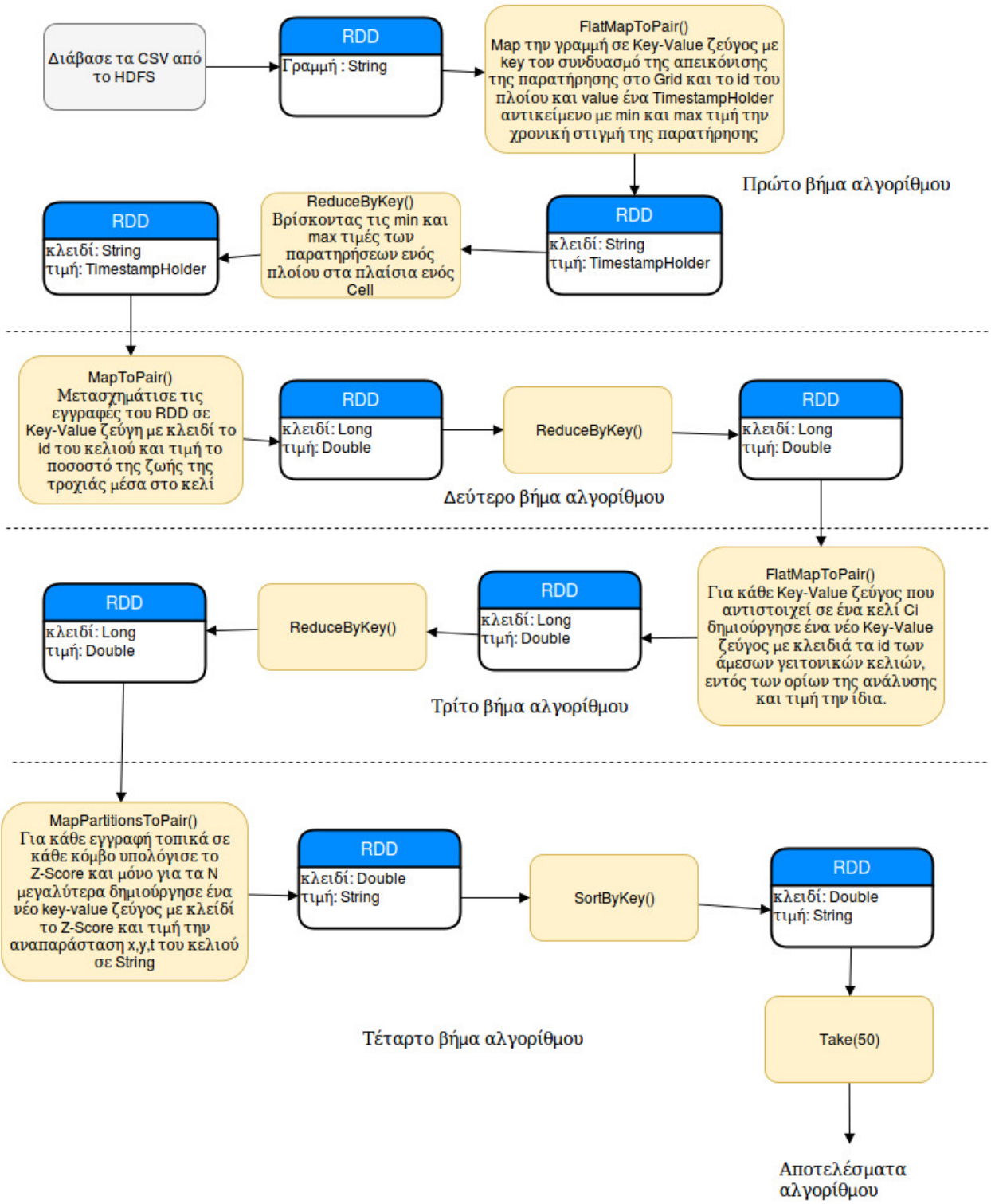
Εικόνα 9: Αναπαράσταση απόδοσης αξίας τροχιάς στα κελιά που διανύει (Trajectory-Aware Algorithm)

Μετά από αυτό ο αλγόριθμος συνεχίζει σαν τον προηγούμενο. Οι επόμενες γραμμές είναι αφιερωμένες στον ψευδοκώδικα και το Διάγραμμα Ροής του αλγορίθμου.

Algorithm 3 Trajecotry-Aware algorithm

```
1: procedure TRAJECTORYAWARE
2: trajectoryRDD = filesOnHDFS.flatMapToPair(p =>
3:     emit new pair((getCellId(p)-p.id).toString(),
4:     new TimestampHolder(p.t))
5: ).reduceByKey(p1, p2 => findMinAndMax(p1.v, p2.v))
6: gridRDD = trajectoryRDD.mapToPair(tr =>
7:     emit new pair ((getCellId(tr.id),
8:     calculateTrajectoryValueOnCell(tr.v))
9: ).reduceByKey(tr1, tr2 => emit tr1.v + tr2.v)
10: gridRDD.forEach(ci => update accumulators)
11: neighborRDD = gridRDD.flatMapToPair(ci =>
12:     N(ci) = getDirectNeighborIds(ci)
13:     for each j in N(ci) do
14:         emit new pair(j, xi)
15:     end for
16: ).reduceByKey(x1, x2 => emit x1 + x2)
17: scoresRDD=neighborRDD.mapPartitionsToPair(ci=>
18:     for each local ci in neighborRDD do
19:         list.add(new pair(ci, G*i))
20:         sort list and keep only the top-k pairs
21:     end for
22:     emit list)
23: sort scoresRDD and return the top-k cells
24: end procedure
```

Αλγόριθμος 3: Trajectory-Aware Αλγόριθμος



Εικόνα 10: Διάγραμμα Ροής Trajectory-Aware Algorithm

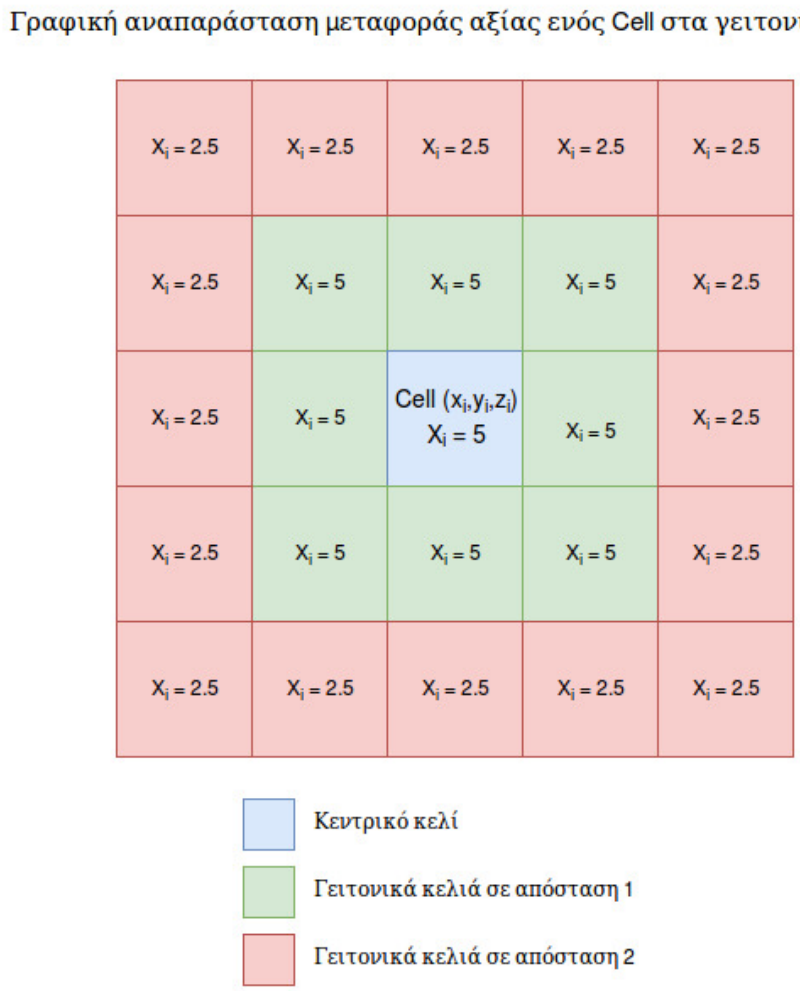
5.2.7 N-Neighbor Algorithm

Ο τρίτος και τελικός αλγόριθμος αποτελεί μία πιο εξελιγμένη μορφή του δεύτερου. Όλα τα βήματα γίνονται ακριβώς όπως και στον δεύτερο με την μόνη διαφορά ότι στο βήμα που δημιουργούνται τα ζευγάρια κλειδιών-τιμών που αντιπροσωπεύουν την αξία που θα προστεθεί στα γειτονικά κελιά δεν λαμβάνονται υπόψιν μόνο οι άμεσοι γείτονες του κεντρικού κελιού. Σε αυτόν τον αλγόριθμο έχει εισαχθεί μία νέα μεταβλητή η οποία επιτρέπει στα κελιά να μεταφέρουν μέρος της αξίας τους στα γειτονικά τους. Αναφέρθηκε η φράση “μέρος της αξίας” ενός κελιού διότι η αξία που μεταδίδεται από ένα κελί σε ένα γειτονικό του φθίνει ανάλογα με την απόσταση των κελιών. Το βάρος της αξίας που τελικά θα μεταφέρει ένα κελί σε ένα άλλο υπολογίζεται βάσει του ακόλουθου τύπου

$$w_{i,j}=\frac{1}{a^{cd}}$$

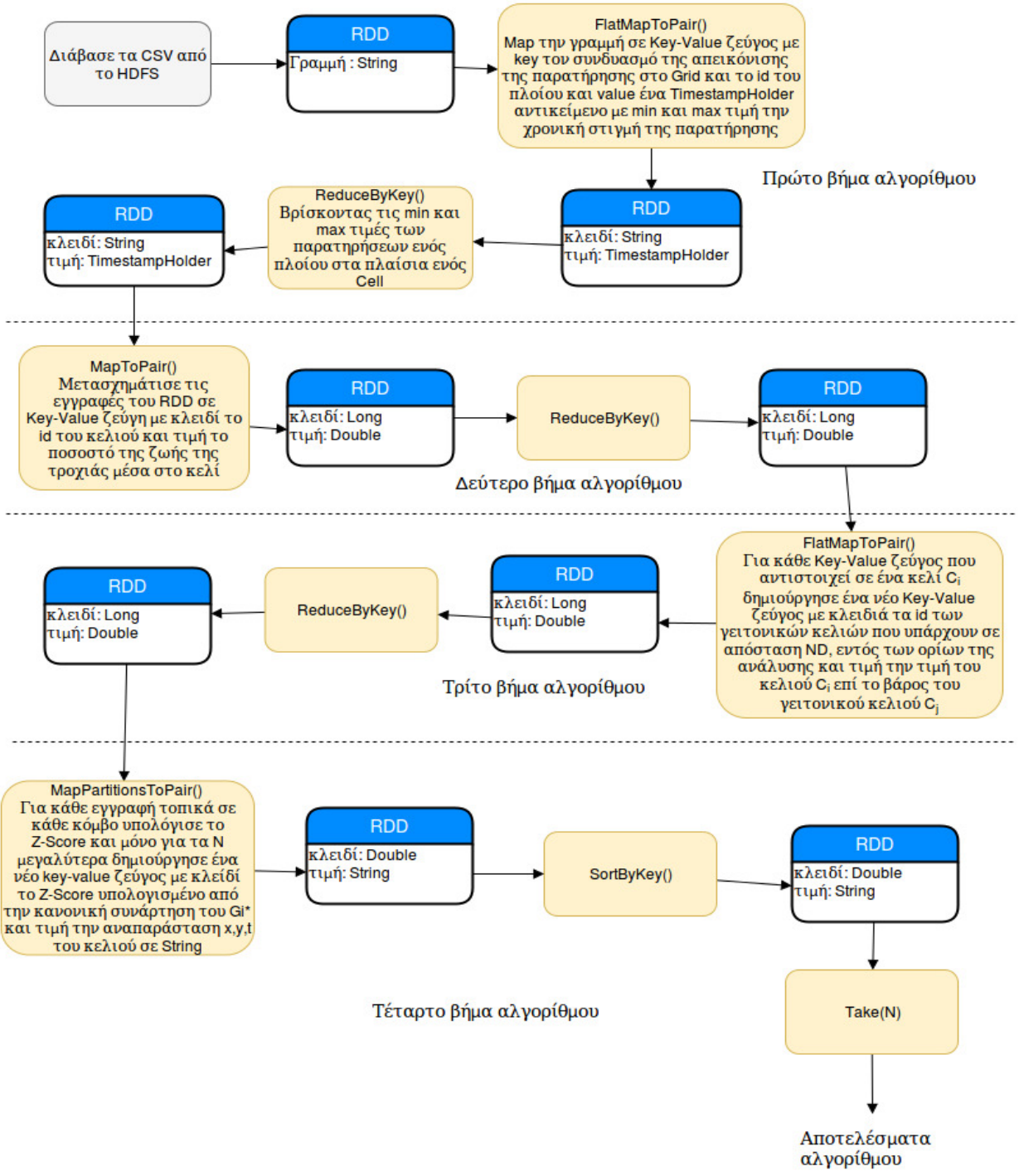
όπου το a είναι μία μεταβλητή μικρότερη της μονάδας η οποία επηρεάζει το ποσοστό μετάδοσης της αξίας ενός κελιού στα γειτονικά του και το cd η απόσταση των κελιών c_i και c_j. Έτσι οι αλλαγές αλγόριθμο επηρεάζουν το MapReduce που υπολογίζει την αξία των γειτόνων και το σημείο που υπολογίζεται η G_i^* καθώς πλέον χρησιμοποιείται ο τύπος (1).

Πιο αναλυτικά η Map φάση του προαναφερθέντος MapReduce μετασχηματίζεται έτσι ώστε να συνυπολογίζει το βάρος της απόστασης δύο κελιών c_i και c_j κατά την δημιουργία των νέων ζευγών κλειδιών-τιμών. Αξίζει να σημειωθεί ότι αυτή η διαδικασία αν και παράγει πολλά νέα στοιχεία μέσα στο RDD, εκτελείται πολύ γρήγορα στο Spark και επηρεάζει αμελητέα τον συνολικό χρόνο εκτέλεσης του αλγορίθμου, όπως θα φανεί στο επόμενο κεφάλαιο όπου εμφανίζονται τα αποτελέσματα της ανάλυσης. Επίσης επιτρέπει την δημιουργία περισσότερων ζευγών καθώς όσο μεγαλύτερη είναι η δοθείσα απόσταση τόσο περισσότερα ζεύγη θα δημιουργηθούν. Για παράδειγμα αν η απόσταση που έχει δοθεί είναι 2 τότε θα δημιουργηθούν 124 νέα ζεύγη, Ο τύπος υπολογισμού είναι $(1+2*nd)^3-1$ όπου το nd είναι η απόσταση που έχει δοθεί από τον χρήστη. Αφού γίνει το reduce και υπολογιστεί το $\sum_{j=0}^{N(c_i)} x_j$ ακολουθεί η επόμενη Map διεργασία η οποία υπολογίζει το G_i^* με τον τύπο (1) αυτή την φορά. Έτσι ο υπολογισμός του Z-score ολοκληρώνεται με ακόμα μεγαλύτερη ακρίβεια και τα αποτελέσματα που παράγονται από τον αλγόριθμο είναι πιο αξιόπιστα. Στην εικόνα που ακολουθεί παρουσιάζεται γραφικά η μεταφορά της αξίας x_i ενός κελιού c_i στα γειτονικά του $N(c_i)$ με α=2.



Εικόνα 11: Γραφική αναπαράσταση μεταφοράς αξίας ενός Cell στα γειτονικά του για a=2

Στην συνέχεια ο αλγόριθμος εκτελεί κανονικά το βήμα του Reduce. Όταν φτάσει η στιγμή να υπολογιστεί η Getis-Ord συνάρτηση υπολογίζονται τα $\sum_{j=0}^{N(c_i)} w_{i,j}$, $\left(\sum_{j=0}^{N(c_i)} w_{i,j}\right)^2$ και $\sum_{j=0}^{N(c_i)} \left(w_{i,j}\right)^2$ τα οποία είναι απαραίτητα για τον υπολογισμό της συνάρτησης (1). Αυτό μπορεί να επιτευχθεί εύκολα καθώς το κάθε κελί γνωρίζει ποια είναι τα γειτονικά του και άρα μπορεί να υπολογίσει τα βάρη των αξιών που έχει λάβει από τα γειτονικά κελιά εκ νέου. Ίσως αυτό το βήμα να μπορεί να δεχθεί βελτιστοποίηση και να μειώνει αρκετά την αποδοτικότητα του αλγορίθμου για ένα σεβαστό αριθμό $N(c_i)$. Όταν τα υπολογίσει μπορεί τελικά να εκτιμήσει το Z-Score του κελιού και μετέπειτα να τα κατατάξει σε φθίνουσα σειρά. Στην εικόνα 16 παρουσιάζεται το Διάγραμμα Ροής του 3ου αλγορίθμου.



Εικόνα 12: Διάγραμμα Ροής N-Neighbor Algorithm

Algorithm 4 N-Neighbor algorithm

```
1: procedure NNEIGHBOR
2: trajectoryRDD = filesOnHDFS.flatMapToPair(p =>
3:     emit new pair((getCellId(p)-p.id).toString(),
4:     new TimestampHolder(p.t))
5: ).reduceByKey(p1, p2 => findMinAndMax(p1.v, p2.v))
6: gridRDD = trajectoryRDD.mapToPair(tr =>
7:     emit new pair ((getCellId(tr.id),
8:     calculateTrajectoryValueOnCell(tr.v))
9: ).reduceByKey(tr1, tr2 => emit tr1.v + tr2.v)
10: gridRDD.forEach(ci => update accumulators)
11: neighborRDD = gridRDD.flatMapToPair(ci =>
12:     N(ci) = getNeighborIds(ci, nd)
13:     for each j in N(ci) do
14:         emit new pair(j, xi * wi,j)
15:     end for
16: ).reduceByKey(x1, x2 => emit x1 + x2)
17: scoresRDD=neighborRDD.mapPartitionsToPair(ci=>
18:     for each local ci in neighborRDD do
19:         list.add(new pair(ci, G*i))
20:         sort list and keep only the top-k pairs
21:     end for
22:     emit list)
23: sort scoresRDD and return the top-k cells
24: end procedure
```

Αλγόριθμος 4: N-Neighbor Αλγόριθμος

6. Πειραματικό μέρος

Σε αυτό το κεφάλαιο παρουσιάζονται τα ευρήματα του πειραματικού μέρους της εργασίας. Τα ευρήματα μπορούν να χωριστούν σε δύο κατηγορίες και είναι τα αποτελέσματα της hotspot ανάλυσης και η συνολική απόδοση του αλγορίθμου ανάλογα τα διαφορετικά ορίσματα που μπορεί να του εισάγει ο χρήστης. Για τον αλγόριθμο BigCAB θα παρουσιαστούν μονάχα αποτελέσματα της απόδοσης του αλγορίθμου όπως εμφανίστηκαν στην πρώτη του παρουσίαση. Για τους υπόλοιπους αλγορίθμους θα δοθεί περισσότερη έμφαση και πάλι στην αποδοτικότητα τους και θα αναφερθούν κάποιες παρατηρήσεις από τις στατιστικά πιο σημαντικές που βρέθηκαν με τον N-Neighbor αλγόριθμο. Λόγω έλλειψης του κατάλληλου μηχανισμού εικονοποίησης αποφασίστηκε να μη δοθεί μεγάλη έμφαση στα αποτελέσματα της ανάλυσης αν και είναι σημαντικό σημείο των ικανοτήτων της.

6.1 Αποτελέσματα ανάλυσης BigCAB

Τα πειράματα για τον αλγόριθμο BigCAB έχουν προηγηθεί σε διαφορετικό cluster. Το συγκεκριμένο cluster περιείχε 16 κόμβους. Οι πρώτοι 8 κόμβοι είχαν 32 GB RAM, 2 CPUs με συνολικά 8 πυρήνες στα 2.6 Ghz και 2 σκληρούς δίσκους, ο καθένας, που το άθροισμα τους έφτανε στα 5 TB. Οι επόμενοι 4 κόμβοι είχαν 128 GB RAM, 2 CPUs με 12 πυρήνες στα 2.6 Ghz και 4 σκληρούς δίσκους, ο καθένας, με συνολική χωρητικότητα κοντά στα 8 TB. Οι υπόλοιποι κόμβοι είχαν 128 GB RAM 2 CPUs με 16 πυρήνες στα 2.6 Ghz, και 4 δίσκους για τον HDFS, με συνολική χωρητικότητα 8 TB. Κάθε κόμβος του cluster λειτουργούσε ως DataNode και NodeManager ενώ ένας ήταν ο MasterNode που λειτουργούσε και ως NameNode και ResourceManager.

Τα δεδομένα που χρησιμοποιήθηκαν για την ανάλυση ήταν 12 CSV αρχεία που περιέγραφαν την κίνηση των ταξί στην Νέα Υόρκη για το έτος 2009 (ένα CSV αρχείο για κάθε μήνα του έτους) και συνολικά ήταν περίπου 24GB πληροφορίας. Τα όρια της ανάλυσης ορίστηκαν ως ένα νοητό παραλληλόγραμμο το οποίο περιβάλλει το μεγαλύτερο τμήμα της πόλης της Νέας Υόρκης και πιο συγκεκριμένα ήταν το 40.5N - 40.9N, 73.7W – 74.25W.

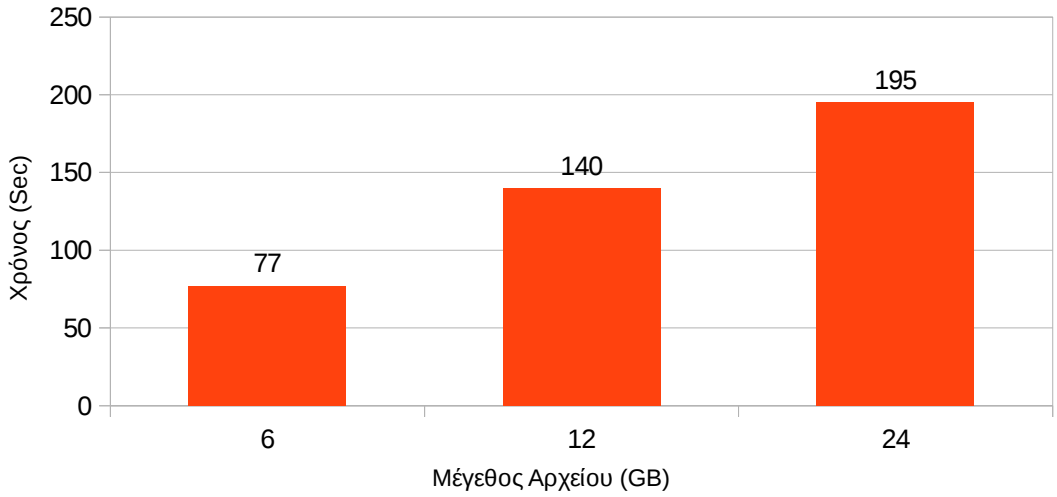
Για την ανάλυση το grid που ορίστηκε ήταν σταθερό και είχε $\partial d = 0.001$ και $\partial t = 0.1$ το οποίο είναι ίσο με περίπου 2.5 ώρες της ημέρας. Αυτές οι τιμές δημιουργούσαν περίπου 800 εκατομμύρια κελιά μέσα στο grid.

Τα πειράματα επικεντρώθηκαν στην παρατήρηση της απόδοσης του αλγορίθμου όταν αλλάζει το μέγεθος του grid και το συνολικό ύψος των δεδομένων που οφείλει να επεξεργαστεί. Στον παρακάτω πίνακα δίνεται η πειραματική διάταξη του BigCAB.

	Τιμή 1	Τιμή 2	Κεντρικη Τιμή
Δεδομένα (GB)	6	12	**24**
Αριθμός Cells	12 K	100 M	**800 M**

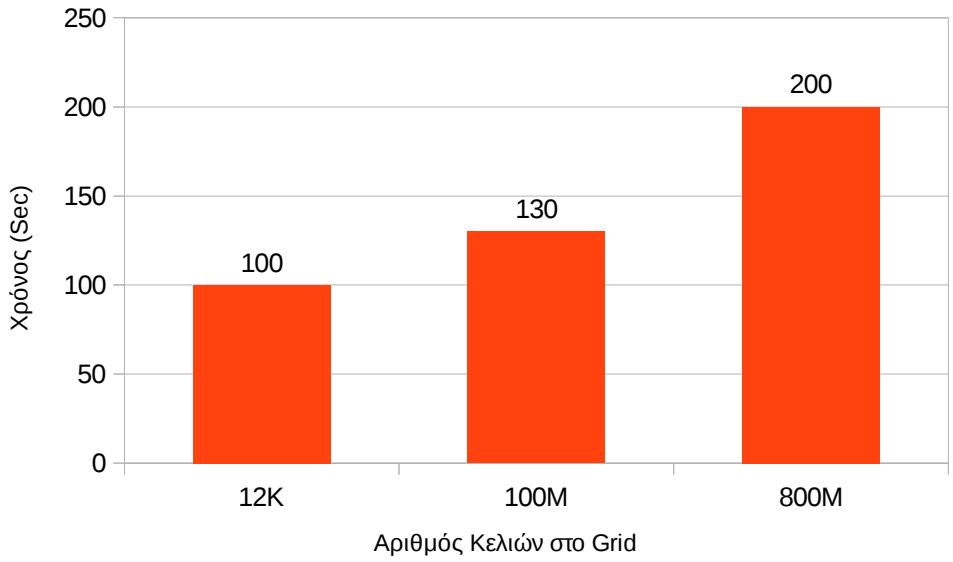
Πίνακας 1: Πειραματική Διάταξη BigCAB

Για 6 GB δεδομένων ο αλγόριθμος χρειάστηκε μόλις 77 δευτερόλεπτα για να ολοκληρώσει την εκτέλεση του, ενώ για 24 GB δεδομένων περίπου 200. Αυτό δείχνει ότι με έναν τετραπλασιασμό των δεδομένων ο συνολικό χρόνος που χρειάστηκε για εκτέλεση δεν τετραπλασιάστηκε. Αυτό δείχνει ότι ο αλγόριθμος BigCAB μπορεί να παρουσιάσει πολύ καλύτερα αποτελέσματα σε μεγάλο όγκο δεδομένων.



Διάγραμμα 1: Διάγραμμα απόδοσης BigCAB (Μέγεθος Αρχείου - Χρόνος)

Στο επόμενο σκέλος των πειραμάτων μεταβλήθηκε το μέγεθος των κελιών. Όσο περισσότερα κελιά περιέχει το grid της ανάλυσης τόσο πιο μεγάλος και ο αριθμός των ζευγαριών κλειδιών-τιμών που έχει να διαχειριστεί ο αλγόριθμος. Έτσι τα βήματα MapReduce αργούν περισσότερο να εκτελεστούν. Ωστόσο όσο περισσότερα είναι τα κελιά της ανάλυσης τόσο καλύτερα αποτυπώνονται τα γεγονότα και μπορεί ένας ερευνητής να αντλήσει ακόμα περισσότερες πληροφορίες από τα αποτελέσματα της ανάλυσης. Στα Διαγράμματα 1 και 2 παρουσιάζεται η γραφικές παράστασεις της αποδοτικότητας του BigCAB αλγορίθμου.



Διάγραμμα 2: Διάγραμμα απόδοσης BigCAB (Μέγεθος Grid - Χρόνος)

6.3 Αποτελέσματα Ανάλυσης Προεκτάσεων BigCAB

Τα πειράματα για τους Simple-Mind Algorithm, Trajectory-Aware και N-Neighbor εκτελέστηκαν στο Okeanos [23], σε cluster το οποίο στήθηκε στα πλαίσια της εργασίας. Το περιβάλλον απαρτίζεται από δέκα κόμβους. Ο κεντρικός κόμβος (Master Node) είχε 6 GB RAM. Όλοι οι υπόλοιποι κόμβοι (Slave Nodes) είχαν 8 GB RAM. Επίσης όλοι οι κόμβοι είχαν 4 CPU cores και 100 GB σκληρό δίσκο με λειτουργικό Ubuntu. Σε όλους τους κόμβους του συστήματος εγκαταστάθηκε HDFS και Yarn ενώ στον κεντρικό και οι βιβλιοθήκες του Spark. Ακολουθήθηκε η Spark on Yarn αρχιτεκτονική. Κάθε κόμβος του cluster λειτουργούσε ως DataNode και NodeManager ενώ ένας ήταν ο MasterNode που λειτουργούσε και ως NameNode και ResourceManager ενώ λειτουργούσε ως client όταν έτρεχαν οι Spark εφαρμογές. Ο Master κάθε εκτέλεσης επιλεγόταν τυχαία από τα DataNodes του HDFS και διαχειριζόταν την πορεία της εκτέλεσης του Spark προγράμματος.

6.4 Αποτελέσματα Ανάλυσης Simple-Mind Algorithm

Η υλοποίηση του Simple-Mind Algorithm όπως έχει ειπωθεί είναι αρκετά αφελής (naive) και τα αποτελέσματα που παράγει όχι και τόσο αξιόπιστα, όμως έχει αξία να παρουσιαστεί το πως ομαλοποιήθηκαν τα δεδομένα με το κόστος της αύξησης της υπολογιστικής ισχύς. Ο παρακάτω πίνακας περιγράφει την πειραματική διάταξη που χρησιμοποιήθηκε.

	Ελάχιστη 1	Κεντρικη Τιμή	Μέγιστη 2
∂d	0.05	**0.1**	0.5
∂t	0.05	**0.1**	0.5

Πίνακας 2: Πειραματική Διάταξη Simple-Mind

Συνολικά πέντε πειράματα πραγματοποιήθηκαν για να ικανοποιηθούν οι ανάγκες της ανάλυσης με βάση τον παραπάνω πίνακα. Η κεντρική τιμή αποτέλεσε την βάση της ανάλυσης και σε κάθε ανάλυση μπορούσε να αλλάζει μόνο ένα από τα παραπάνω ορίσματα. Τα όρια της ανάλυσης ορίστηκαν ως ένα νοητό παραλληλόγραμμο το οποίο περιβάλλει το Αιγαίο πέλαγος.

Στα πλαίσια του πειράματος από τις 1.934 δισεκατομμύρια εγγραφές που υπήρχαν μέσα στο σύνολο των δεδομένων στην ανάλυση χρησιμοποιήθηκαν περίπου 600 εκατομμύρια. Οι υπόλοιπες ήταν εκτός των ορίων της ανάλυσης. Αξίζει να σημειωθεί ότι οι χρόνοι εκτέλεσης μπορεί να κυμαίνονται ελάχιστα εξαιτίας της μεγάλης μεταφοράς πληροφοριών από κόμβο σε κόμβο κατά την διάρκεια του shuffling και του reduce. Τα Map βήματα προφανώς είναι πολύ πιο γρήγορα αφού επεξεργάζονται την πληροφορία που έχει ο κάθε κόμβος τοπικά.

Επίσης ο αλγόριθμος χωρίστηκε, όπως φαίνεται στην εικόνα 12, σε τέσσερις φάσεις (P1, P2, P3 και P4 στον πίνακα). Για την διευκόλυνση του ο αναγνώστης μπορεί να ανατρέξει πίσω στο προηγούμενο κεφάλαιο όπου και παρουσιάζεται η δομή του Simple-Mind Algorithm.

Τέλος υπάρχει ένα βήμα το οποίο δεν αναφέρεται καθόλου και ήταν πάντα σταθερό ή με αμελητέες διακυμάνσεις. Αυτό είναι το βήμα του υπολογισμού κάποιων στατιστικών και διαρκούσε περίπου 30 δευτερόλεπτα.

Ακολουθούν δύο υποκεφάλαια τα οποία περιγράφουν την αποδοτικότητα για κάθε ένα από τα παραπάνω ορίσματα. Σε κάθε υποκεφάλαιο μεταβάλλεται μόνο ένα όρισμα και τα άλλα παραμένουν σταθερά.

6.4.1 Αποτελέσματα με μεταβλητότητα στο πεδίο του χώρου

Το πεδίο του χώρου εκφράζεται από το ∂d και ως κεντρική τιμή έχει οριστεί το 0.1 και εκφράζει γεωγραφικές μοίρες. Οι δύο τιμές που επιλέχθηκαν για τον έλεγχο της αποδοτικότητας του αλγορίθμου είναι οι 0.5 και 0.05 όπως παρουσιάστηκαν και στον παραπάνω πίνακα.

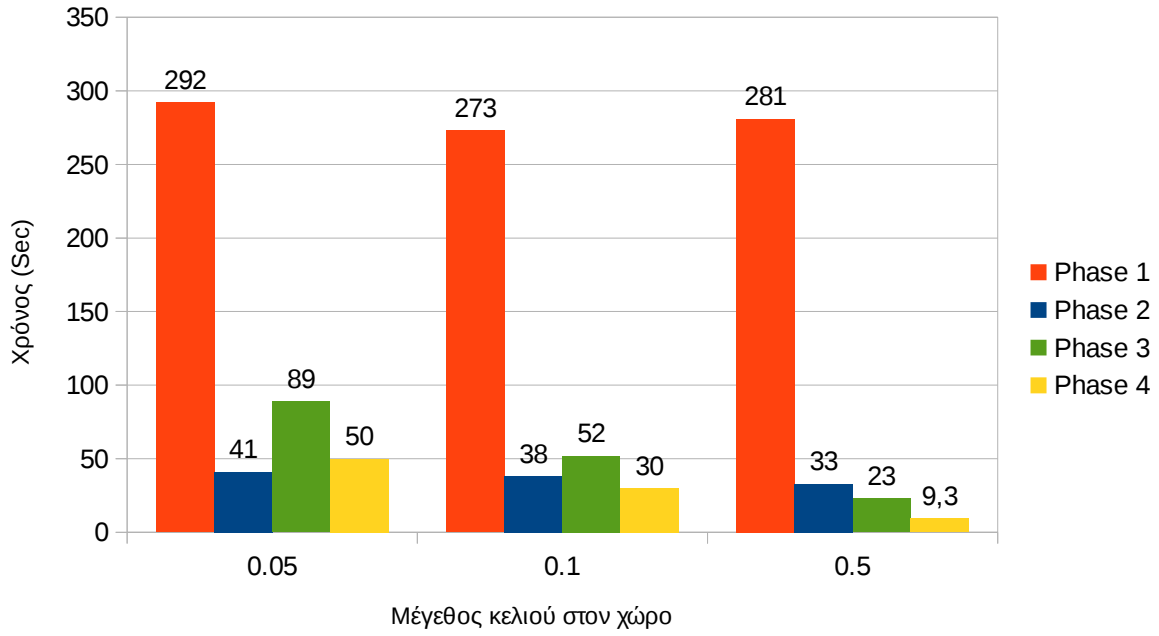
Τα αποτελέσματα των χρόνων εκτέλεσης γι’αυτές τις 4 φάσεις είναι σε second. Επίσης παρουσιάζονται χρήσιμα στατιστικά όπως πόσα κελιά δημιουργήθηκαν (No C_i), ο συνολικός αριθμών των κελιών που είχαν τιμή X_i (Count 1) και το συνολικός αριθμός των κελιών που απέκτησαν τιμή μετά την μεταφορά της αξίας των γειτονικών κελιών (Count 2).

	P1	P2	P3	P4	No C_i	Count 1	Count 2
$\partial d=0.05$	292 s	41 s	89 s	50 s	73.8 M	6.08 M	25.7 M
$\partial d=0.1$	273 s	38 s	52 s	30 s	18.7 M	2.98 M	9.51 M
$\partial d=0.5$	281 s	33 s	23 s	9.3 s	790 K	416 K	642 K

Πίνακας 3: Αποτελέσματα πρώτης σειράς πειραμάτων Simple-Mind Αλγορίθμου

Από τον παραπάνω πίνακα γίνεται αντιληπτό ότι οι χρόνοι εκτέλεσης δεν αυξάνονται γραμμικά με τα δεδομένα που βρίσκονται και αυτό είναι αποτέλεσμα της παράλληλης επεξεργασίας του αλγόριθμου. Το πρώτο βήμα θεωρείται περίπου ίδιο καθώς εκεί έχει να διαβάσει και τα αρχεία από το HDFS. Επίσης εάν ληφθεί υπόψιν ότι ο αριθμός Count 1 επηρεάζει την P2 και την P3 και ο αριθμός Count 2 την P3 και την P4 τα αποτελέσματα φαίνεται να καθιστούν τον αλγόριθμο αρκετά αποδοτικό ακόμα και σε μεγαλύτερο όγκο δεδομένων. Επίσης κάτι το οποίο δεν είναι αρκετά εμφανές είναι ότι επειδή η παράμετρος ∂d επηρεάζει δύο διαστάσεις ενός cell ταυτόχρονα, μειώνοντας την συγκεκριμένη παράμετρο στο μισό ο αριθμός τον παραγόμενων cells τετραπλασιάζεται. Αυτό το φανερώνουν και οι διάφορες τιμές των No C_i.

Στο ακόλουθο διάγραμμα φαίνεται σχηματικά η αποδοτικότητα του αλγορίθμου για κάθε διαφορετικό όρισμα.



Διάγραμμα 3: Διάγραμμα απόδοσης Simple-Mind (Μέγεθος Grid - Χρόνος)

6.4.2 Αποτελέσματα με μεταβλητότητα στο πεδίο του χρόνου

Για την δεύτερη σειρά πειραμάτων μεταβλήθηκε το ∂t με τιμές 0.1, 0.05 και 0.5 όπου αυτά μεταφράζονται σε περίπου 2.5 ώρες, 1.25 περίπου ώρες και 12 ώρες αντίστοιχα. Ο ακόλουθος πίνακας, παρόμοιος με τον παραπάνω, δείχνει τα αποτελέσματα της αποδοτικότητας του αλγορίθμου. Σε αυτή την περίπτωση η αποδοτικότητα της ανάλυσης φαίνεται να επηρεάζεται όπως και στα πειράματα που έγιναν για το ∂d και αυτό γιατί και σε αυτή την περίπτωση επηρεάζεται το μέγεθος του κελιού.

	P1	P2	P3	P4	No C_i	Count 1	Count 2
$\partial t=0.05$	301 s	37 s	70 s	47 s	37.5 M	4.16 M	16.3 M
$\partial t=0.1$	273 s	38 s	52 s	30 s	18.7 M	2.98 M	9.51 M
$\partial t=0.5$	290 s	38 s	35 s	16 s	3.75 M	1.2 M	2.38 M

Πίνακας 4: Αποτελέσματα δεύτερης σειράς πειραμάτων Simple-Mind Αλγορίθμου

Διάγραμμα αποδοτικότητας δεν θα εμφανιστεί σε αυτό το σημείο καθώς ο αλγόριθμος συμπεριφέρεται με τον ίδιο ακριβώς τρόπο άρα μία τέτοια συμπεριφορά θεωρείται ήδη γνωστή.

6.5 Αποτελέσματα Ανάλυσης Trajectory-Aware και N-Neighbor algorithm

Οι υλοποιήσεις των 2 παραπάνω αλγορίθμων είναι κοινές με την μόνη διαφορά ότι ο N-Neighbor έχει την ικανότητα να λαμβάνει υπόψιν του έμμεσους γείτονες ενός κελιού για τον υπολογισμό του Getis-Ord G_i^* στατιστικού. Έτσι μπορούμε εύκολα να μελετήσουμε και τον Trajectory-Aware μέσω του N-Neighbor αλγορίθμου, προφανώς για τιμές nd ίσες με μονάδα. Στον παρακάτω πίνακα παρουσιάζεται η πειραματική διάταξη που χρησιμοποιήθηκε. Για να ικανοποιηθούν οι ανάγκες της ανάλυσης έγιναν συνολικά επτά πειράματα με την κεντρική τιμή να αποτελεί την βάση της ανάλυσης και σε κάθε ανάλυση να αλλάζει μόνο ένα από τα παρακάτω ορίσματα. Τα όρια της ανάλυσης και σε αυτή την περίπτωση ορίστηκαν ως ένα νοητό παραλληλόγραμμο το οποίο περιβάλλει το Αιγαίο πέλαγος.

Θα ακολουθηθεί η ίδια λογική με τον Simple-Mind Algorithm οπότε το κεφάλαιο αυτό θα χωριστεί σε τρεις ενότητες. Μία για κάθε τιμή που θα μεταβάλλεται. Το ξεχωριστό αυτής της ενότητας είναι το nd όρισμα, το οποίο υποδεικνύει την μέγιστη απόσταση δύο γειτονικών κελιών στα οποία επιτρέπεται να προσμετρηθεί η μεταξύ τους αξία. Η κεντρική τιμή του nd γι'αυτή την ανάλυση είναι το 2.

	Ελάχιστη 1	**Κεντρικη Τιμή**	**Μέγιστη 2**
∂d	0.05	**0.1**	0.5
∂t	0.05	**0.1**	0.5
nd	1	**2**	3

Πίνακας 5: Πειραματική Διάταξη N-Neighbor Αλγορίθμου

6.5.1 Αποτελέσματα με μεταβλητότητα στο πεδίο του χώρου

Τα αποτελέσματα τα οποία βγήκαν από την πρώτη σειρά φαίνονται στον παρακάτω πίνακα. Όπως είναι κατανοητό η υπολογιστική ισχύς που χρειαζόταν ο συγκεκριμένος αλγόριθμος για να εκτελεστεί ήταν αρκετά μεγαλύτερη από αυτή του Simple-Mind Algorithm. Σε αυτό το σημείο αξίζει να εισαχθεί μία ακόμα μετρική η **Count emitted** η οποία προσμετράει πόσα νέα ζεύγη κλειδιών-τιμών δημιουργήθηκαν κατά την διάρκεια της P3 για να μεταφέρουν την αξία τους σε γειτονικά κελιά. Ο αριθμός είναι αρκετά μεγάλος για nd = 2 όπως φαίνεται και στα αποτελέσματα της ανάλυσης.

Πιο συγκεκριμένα απ'ότι φαίνεται ο συγκεκριμένος αριθμός επηρεάζει κατά πολύ την φάση P3 και P4. Την P3 γιατί είναι αυτή η οποία τα παράγει στην Map φάση και έπειτα τα κάνει Reduce αλλά και την P4 εμμέσως καθώς πολλαπλασιάζει το Count 2. Βέβαια όλα αυτά επηρεάζονται ταυτόχρονα και από το μέγεθος του κελιού. Όταν τα κελιά είναι αρκετά μεγάλα και λίγα μέσα στον χώρο της ανάλυσης τότε αυξάνεται η πιθανότητα να "ακουμπάνε" τα όρια της ανάλυσης και άρα να μην υπάρχουν γείτονες ώστε να μεταφέρουν την αξία τους.

Αυτό μπορεί να φανεί για $\partial d = 0.5$ όπου το count emitted είναι μόλις 39.3 εκατομμύρια ωστόσο σχετικά μικρός σε σχέση με τα άλλα πειράματα που διεξήχθησαν. Επίσης αυτό μπορεί να φανεί και από την πληρότητα του grid σε πληροφορία. Όταν το Count 2 προσεγγίζει το No C_i τότε το grid γεμίζει με πληροφορία από την ανταλλαγή αξίας γειτονικών κελιών. Να υπενθυμιστεί ότι με nd = 2 παράγονται έως και 124 νέα ζεύγη κλειδιών-τιμών από ένα κελί c_j για τα γειτονικά κελιά $N(c_i)$ σε απόσταση 2 το πολύ.

	P1	P2	P3	P4	No C_i	Count 1	Count emitted	Count 2
$\partial d=0.05$	300 s	50 s	364 s	228 s	73.8 M	6.08 M	745 M	38.2 M
$\partial d=0.1$	284 s	41 s	161 s	107 s	18.7 M	2.98 M	357 M	12.3 M
$\partial d=0.5$	275 s	36 s	35 s	20 s	790 K	416 K	39.3 M	712 K

Πίνακας 6: Αποτελέσματα πρώτης σειράς πειραμάτων N-Neighbor Αλγορίθμου

6.5.2 Αποτελέσματα με μεταβλητότητα στο πεδίο του χρόνου

Το πεδίο του χρόνου είναι αλληλένδετο με το μέγεθος του κελιού και αυτό. Οπότε επηρεάζει με παρόμοιο τρόπο την αποδοτικότητα του αλγορίθμου. Τα αποτελέσματα της δεύτερης σειράς πειραμάτων παρουσιάζονται στον παρακάτω πίνακα. Δεν υπάρχει κάποιο νέο εύρημα ως προς αυτή την σειρά πειραμάτων άρα θα ήταν δόκιμο να μην αναλυθεί περαιτέρω κάποιο από αυτά.

	P1	P2	P3	P4	No C_i	Count 1	Count emited	Count 2
$\partial t=0.05$	297 s	43 s	228 s	141 s	37.5 M	4.16 M	497 M	22.9 M
$\partial t=0.1$	284 s	41 s	161 s	107 s	18.7 M	2.98 M	357 M	12.3 M
$\partial t=0.5$	307 s	40 s	77 s	42 s	3.75 M	1.2 M	144 M	2.8 M

Πίνακας 7: Αποτελέσματα δεύτερης σειράς πειραμάτων N-Neighbor Αλγορίθμου

6.5.3 Αποτελέσματα με μεταβλητότητα στην απόσταση γειτόνων

Ίσως τα πιο σημαντικά ευρήματα για αυτόν τον αλγόριθμο υπάρχουν σε αυτό το υποκεφάλαιο. Έχοντας σταθερό το μέγεθος του κελιού αυξομειώνεται η απόσταση nd. Απ'ότι φαίνεται επηρεάζει σημαντικά την αποδοτικότητα του αλγορίθμου ωστόσο με αυτόν τον τρόπο υπολογίζεται το στατιστικό G_i^* με μεγαλύτερη ακρίβεια λαμβάνοντας υπόψιν περισσότερη πληροφορία από το grid.

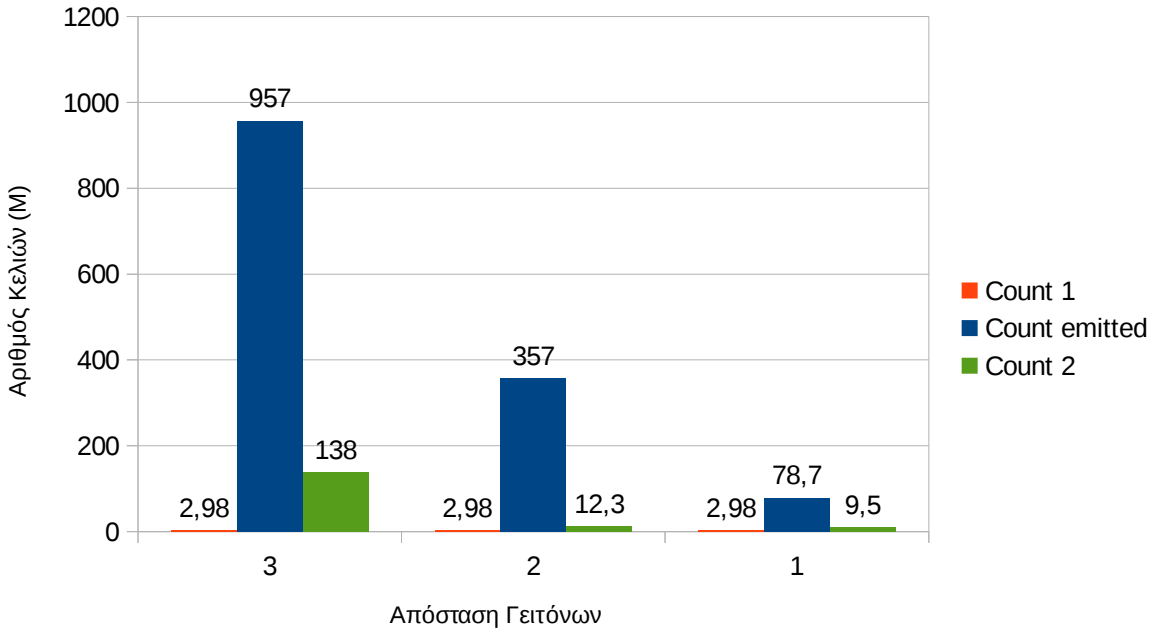
Όπως φαίνεται τα πρώτα βήματα P1 και P2 είναι σχεδόν σταθερά (με αμελητέες διακυμάνσεις που είναι λογικές σε ένα υπολογιστικό περιβάλλον) καθώς το Count 1 παραμένει σταθερό. Το P3 και το P4 που όπως προαναφέρθηκε επηρεάζεται από τα Count emitted και Count 2 αντίστοιχα ανεβάζουν αρκετά τον συνολικό χρόνο εκτέλεσης του αλγορίθμου, ωστόσο και πάλι όχι γραμμικά.

	P1	P2	P3	P4	No C_i	Count 1	Count emited	Count 2
nd = 3	292 s	41 s	393 s	319 s	18.7 M	2.98 M	957 M	13.8 M
nd = 2	284 s	41 s	161 s	107 s	18.7 M	2.98 M	357 M	12.3 M
nd = 1	292 s	42 s	57 s	32 s	18.7 M	2.98 M	78.7 M	9.5 M

Πίνακας 8: Αποτελέσματα τρίτης σειράς πειραμάτων N-Neighbor Αλγορίθμου

Στην γραφική παράσταση που ακολουθεί παρουσιάζει την μεγάλη διαφορά σε ζεύγη κλειδιών-τιμών που παρατηρείται μεταξύ των φάσεων P2 – P3 – P4.

Τέλος άλλη μία μεταβλητή που επηρεάζει απευθείας το P3 και το P4 είναι το ίδιο το nd. Αυτό είναι ένας περιορισμός που έχει ο αλγόριθμος αυτή την στιγμή. Ως έχει δεν μπορεί να επεξεργαστεί δεδομένα για μεγαλύτερο nd από 3 για ένα σχετικά ικανοποιητικό μέγεθος κελιών. Και εκεί θα έπρεπε να εστιάσει μία περαιτέρω εξέλιξη του. Αυτό δεν έχει να κάνει τόσο πολύ με τα νέα key-value pairs που δημιουργεί όσο με τον τρόπο που τα δημιουργεί και αντίστοιχα με τον τρόπο με τον οποίο υπολογίζει το Z-Score.



Διάγραμμα 4: Διάγραμμα απόδοσης N-Neighbor (Απόσταση Γειτόνων – Αριθμός Κελιών)

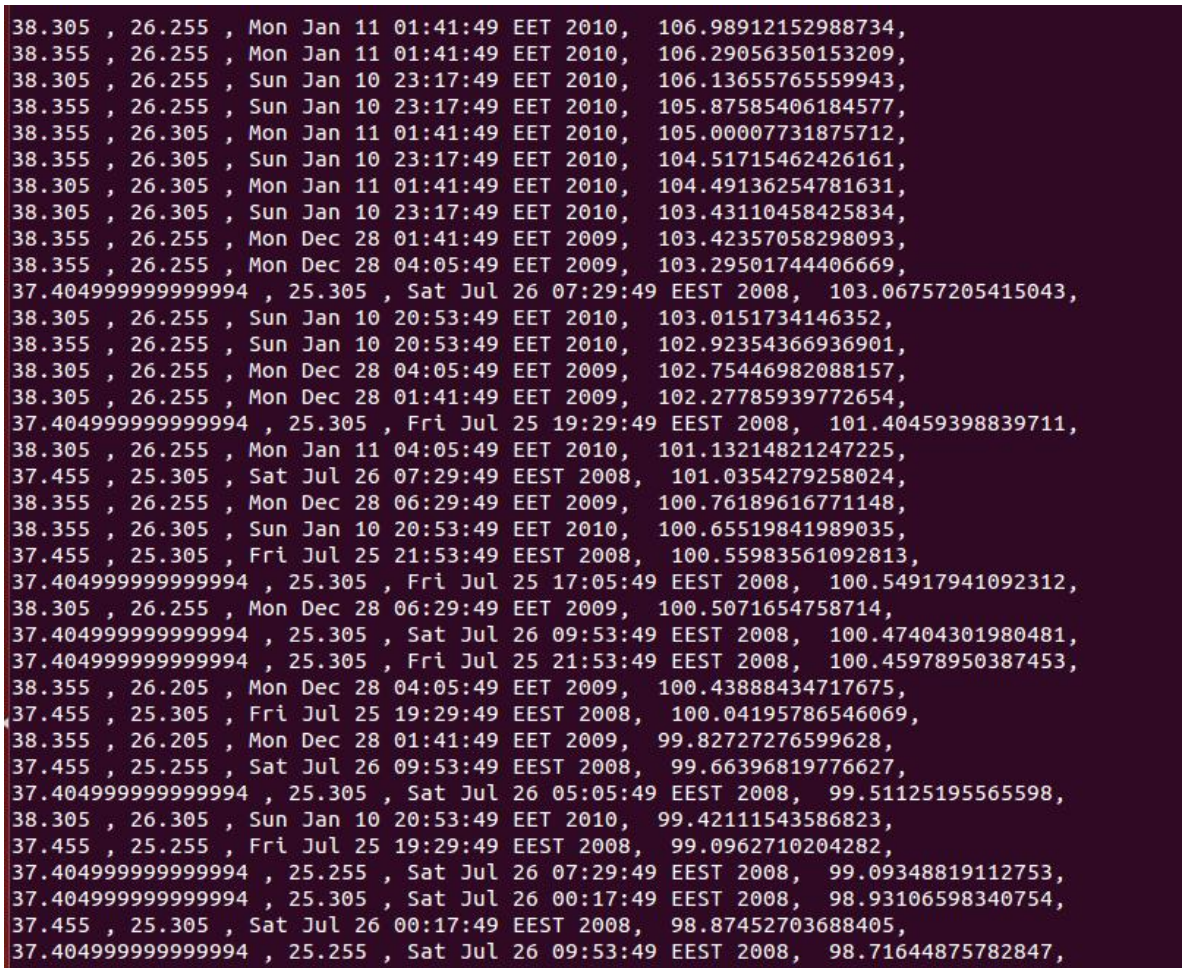
6.6 Top-k παρατηρήσεις

Σε αυτό το σημείο θα γίνει μία αναφορά των αποτελεσμάτων της ανάλυσης έτσι όπως εξήχθησαν από τον αλγόριθμο N-Neighbor για nd = 2 και μέγεθος κελιού 0.05 γεωγραφικές μοίρες στον χώρο και 0.1 ημέρες (περίπου 2.5 ώρες) στον χρόνο. Αρχικά θα πρέπει να αναφερθεί ότι δεν υπήρχε κάποιο εργαλείο για την απεικόνιση των αποτελεσμάτων σε μορφή κατανοητή για τον άνθρωπο οπότε και η έκταση της συγκεκριμένης ενότητας δεν θα είναι μεγάλη.

Τα αποτελέσματα εξάγονται από τον αλγόριθμο και γράφονται σε ένα αρχείο CSV. Η διάταξη που χρησιμοποιείται είναι:

- μέση τεταγμένη κελιού
- μέση τετμημένη κελιού
- μέσος χρόνος κελιού
- Z-Score κελιού

Ενδεικτικά ακολουθεί παράδειγμα από τα παραγόμενα αποτελέσματα.



Εικόνα 13: Αποτελέσματα αλγορίθμου N-Neighbor

Εύκολα μπορεί να διαπιστωθεί ότι υπάρχει μεγάλη συγκέντρωση σε συγκεκριμένες περιοχές. Αυτές ονομάζονται hotspots. Επίσης μπορεί να γίνει αντιληπτό ότι πολλά γειτονικά cells είναι ταξινομημένα μαζί. Αυτό προσδίδει ακόμα μεγαλύτερη αξία στις παρατηρήσεις γιατί δίνει διάρκεια στο φαινόμενο το οποίο περιγράφουν.

Για παράδειγμα οι πρώτες 8 παρατηρήσεις μπορούν να ομαδοποιηθούν μαζί και φαίνεται να προέρχονται από γειτονικά κελιά. Περιγράφουν ένα γεγονός το οποίο είχε να κάνει με μεγάλη συγκέντρωση πλοίων σε μία περιοχή μεταξύ της πόλης Τσεσμέ και της Χίου. Άλλες παρατηρήσεις δείχνουν να εμφανίζονται κοντά σε πολυσύχναστα νησιά του αιγαίου όπως η Μύκονος και η Σύρος. Ενώ άλλα φαίνεται να δημιουργούνται από σταυροδρόμια από ακτοπλοϊκές γραμμές όπως αυτό μεταξύ Πάρου, Νάξου και Μυκόνου.

6.7 Μελλοντικές Βελτιώσεις

Ο αλγόριθμος ως έχει κρίνεται αποδοτικός για ένα πολύ μεγάλο όγκο δεδομένων. Ωστόσο υπάρχουν κομμάτια στον κώδικα τα οποία θα πρέπει να εξελιχθούν για να

αντιμετωπίσουν κάποιους από τους περιορισμούς που προειπώθηκαν σε σχέση με την μέγιστη απόσταση nd που θα δώσει ως όρισμα ο χρήστης. Ένας σημαντικός περιορισμούς είναι η δημιουργία των πολλαπλών ζευγών κλειδιών-τιμών και η Reduce φάση που τα ακολουθεί. Για μεγάλο αριθμό nd η πολυπλοκότητα της εκτέλεσης αυτού του βήματος ανεβαίνει και η αποδοτικότητα μειώνεται.

Μία πιθανή λύση σε αυτό το πρόβλημα θα ήταν η ομαδοποίηση άμεσων και έμμεσων γειτονικών κελιών σε ένα αντικείμενο που τα περικλείει, σε προηγούμενο βήμα της ανάλυσης, ώστε να στέλνονται προς επεξεργασία στον ίδιο κόμβο. Αυτό θα μπορούσε να αποφορτίσει το δίκτυο κατά την Reduce φάση αφού ο κάθε κόμβος θα είχε ένα μέρος από τα δεδομένα για να κάνει το Reduce.

Ένα επιπλέον βήμα που μπορεί να είναι χρονοβόρο και επηρεάζεται άμεσα από το nd είναι η τελευταία MapReduce φάση όπου υπολογίζεται το Z-Score. Σε αυτή την περίπτωση πρέπει να βρεθεί ένας πιο αποδοτικός τρόπος για τον υπολογισμό των βαρών.

8. Επίλογος

Μεγάλος όγκος χώρο-χρονικών δεδομένων και δεδομένων τροχιών δημιουργείται καθημερινά από τροχιές αντικειμένων, μηνύματα και check ins στα μέσα κοινωνικής δικτύωσης και άλλα. Η ανάλυση τέτοιων τύπων δεδομένων λέγεται ότι έχει την δυνατότητα να αποκαλύψει κρυφά μοτίβα και άλλες πληροφορίες οι οποίες πολλές φορές δεν είναι εμφανείς. Έτσι και το Hotspot Analysis είναι μία τεχνική ανάλυσης για την αποτύπωση τέτοιων στατιστικά αξιοσημείωτων φαινομένων συγκεντρωμένα γύρω από μία περιοχή. Πρωτοεμφανίστηκε ως μία μορφή ανάλυσης χωρικών και γεωγραφικών δεδομένων ωστόσο μετέπειτα μεταφέρθηκε και στον τομέα τον χώρο-χρονικών δεδομένων.

Αυτή η διπλωματική επικεντρώθηκε στην μεταφορά της Hot Spot ανάλυσης στον τομέα των χώρο-χρονικών δεδομένων και τον τροχιών θέτοντας νέες διαστάσεις στο πρόβλημα. Πρότεινε αξιοπρεπείς λύσεις για την αποδοτική ανάλυση δεδομένων μεγάλης κλίμακας και δημιούργησε νέες προκλήσεις για την βελτιστοποίηση της ανάλυσης.

Παραπομπές

[1] John R. Mashey (25 April 1998). "Big Data ... and the Next Wave of InfraStress" (PDF). Slides from invited talk. Usenix.

[2] Hilbert, M. (2016), Big Data for Development: A Review of Promises and Challenges. Dev Policy Rev, 34: 135–174. doi:10.1111/dpr.12142

[3] EVANS L. (Ed). The Large Hadron Collider: a Marvel of Technology. CERN and EPFL Press (2009). Chapter 5.6.

[4] Jeffrey Dean, Sanjay Ghemawat:

MapReduce: Simplified Data Processing on Large Clusters. OSDI 2004: 137-150

[5] http://hadoop.apache.org/

[6] Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung:

The Google file system. SOSP 2003: 29-43

[7] https://spark.apache.org/

[8] http://storm.apache.org/

[9]Mazimpaka, Jean Damascène and Timpf, Sabine (2016) "Trajectory data mining: A review of methods and applications," Journal of Spatial Information Science: Iss. 13, 61-99.

[10]The analysis of spatial association by use of distance statistics

A Getis, JK Ord

Geographical analysis 24 (3), 189-206

[11] Moran, P. A. P. (1950). "Notes on Continuous Stochastic Phenomena". Biometrika. ***37*** *(1): 17–23*

[12] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauly, Michael J. Franklin, Scott Shenker, Ion Stoica:

Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. NSDI 2012: 15-28

[13] Liang Hong, Yu Zheng, Duncan Yung, Jingbo Shang, Lei Zou:

Detecting urban black holes based on human mobility data. SIGSPATIAL/GIS 2015: 35:1-35:10

[14] H. Klessig, V. Suryaprakash, O. Blume, A. J. Fehske, and G. Fettweis. A framework enabling spatial analysis of mobile trac hot spots.

[15] http://hive.apache.org/

[16] http://mahout.apache.org/

[17] https://hbase.apache.org/

[18] https://cassandra.apache.org/

[19] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D. Joseph, Randy H. Katz, Scott Shenker, Ion Stoica:

Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center. NSDI 2011

[20] http://sigspatial2016.sigspatial.org/giscup2016/

[21] http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml

[22] https://www.nctr.usf.edu/wp-content/uploads/2011/04/JPT14-1Truong.pdf

[23] https://okeanos.grnet.gr/home/