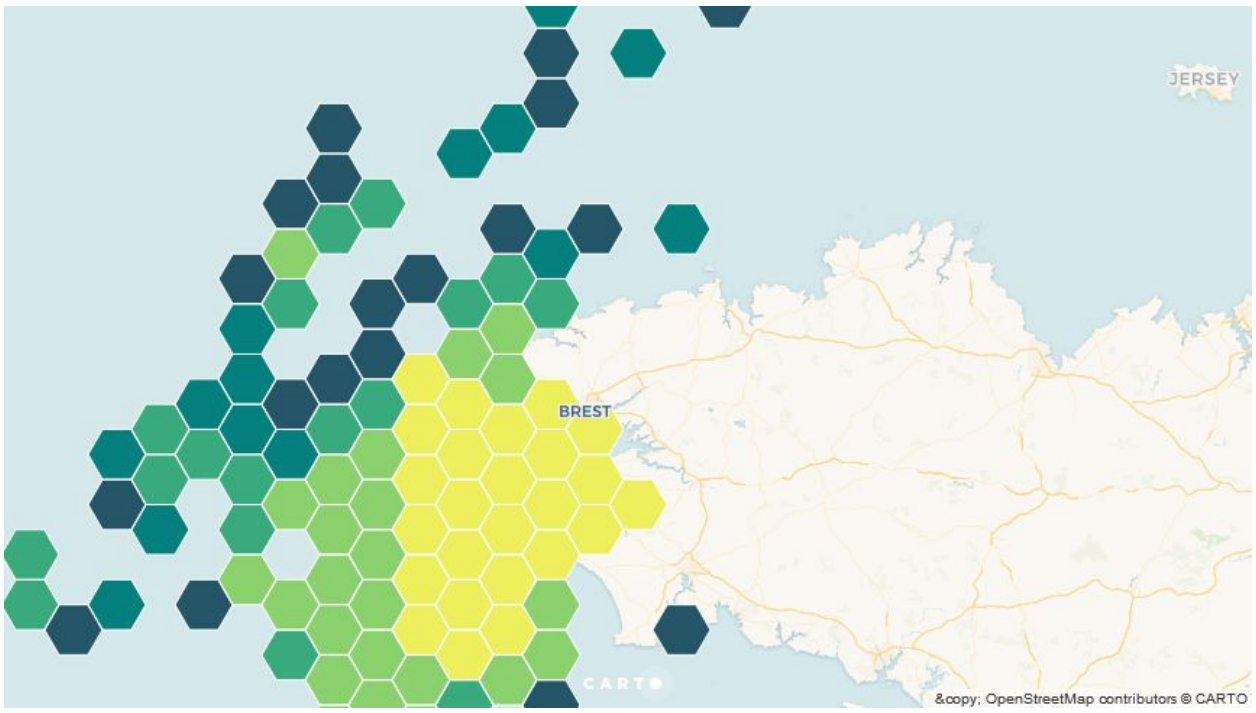


Σύστημα προειδοποίησης συγκρούσεων πλοίων σε Spark Streaming

Εργασία μαθήματος Μεγάλα Δεδομένα και Αναλυτική

Πανεπιστήμιο Πειραιώς

Αμερικάνος Πάρις-Παναγιώτης

ΜΕ 1703

Εισαγωγή

Στην εργασία του μαθήματος Μεγάλα Δεδομένα και Αναλυτική τέθηκε η ελεύθερη επιλογή θέματος βάσει συγκεκριμένου dataset με δεδομένα ναυσιπλοΐας στη θαλάσσια περιοχή βορειοδυτικά της Γαλλίας. Υλοποιήθηκε σύστημα ανίχνευσης, επεξεργασίας, προειδοποίησης και οπτικοποίησης επικείμενων συγκρούσεων πλοίων σε πραγματικό χρόνο βάσει του κατανεμημένου υπολογιστικού πλαισίου μεγάλων δεδομένων Spark Streaming.

Περιγραφή Dataset

Το dataset που υποδείχθηκε για ανάλυση περιλαμβάνει δεδομένα πλοίων συλλεγμένα μέσω του Automatic Identification System (AIS), καθώς και συμπληρωματικά δεδομένα ναυσιπλοΐκής, ναυτιλιακής, γεωγραφικής και μετεωρολογικής φύσης[1] (Ray, 2018). Τα δεδομένα αυτά είναι χωρικά και χρονικά ευθυγραμμισμένα, καλύπτοντας την περίοδο μεταξύ 1/10/2015 και 31/3/2016 και τη θαλάσσια περιοχή της Κέλτικης Θάλασσας, της Μάγχης και του Κόλπου του Biscay. Πιο συγκεκριμένα, περιλαμβάνουν τροχιές και χαρακτηριστικά αναγνώρισης πλοίων με ρυθμό ανανέωσης 10-180 sec, συντεταγμένες λιμένων, σύνορα θαλάσσιων περιοχών, καιρικές και θαλάσσιες συνθήκες, όλα αποθηκευμένα σε υποφακέλους με μορφές CSV, SHP, κά. Η περιοχή κάλυψης είναι περιορισμένη μεταξύ των γεωγραφικών μηκών 45° έως 51° και πλατών -10° έως 0°.

Το αρχείο nari_dynamic.csv που περιέχει τις τροχιές των πλοίων περιλαμβάνει πληροφορίες ανά χρονική στιγμή για τις συντεταγμένες θέσης, την κίνηση και το χαρακτηριστικό κλήσης τους (MMSI). Με βάση το MMSI μπορούν να αναζητηθούν επιπλέον πληροφορίες για κάθε πλοίο από άλλα αρχεία (nari_static.csv, ship_types_list.csv, navigational_status.csv, κά) σχετικά με τεχνικά χαρακτηριστικά του (πχ. μέγεθος, εκτόπισμα, τύπος) και την κατάστασή του (αγκυροβόλι, ακυβέρνητο, εν πλω, κλπ).

Ανάπτυξη προβλήματος

Το dataset καλύπτει με σημαντική λεπτομέρεια τη ΝΔ είσοδο του Καναλιού της Μάγχης, την πιο πολυσύχναστη ναυτιλιακή περιοχή στον κόσμο. Περισσότερα από

[1] https://zenodo.org/record/1167595

500 πλοία διασχίζουν το Κανάλι καθημερινά, από μικρά ιστιοπλοϊκά έως tankers, εγκάρσια και παράλληλα. Λόγω του μεγάλου πλήθους και όγκου των πλοίων, της γεωγραφίας του Καναλιού (33 χμ πλάτος) και της σημασίας του στο παγκόσμιο εμπόριο (Mambra, 2017), η μεγαλύτερη πρόκληση που αντιμετωπίζουν οι διαχειριστές του είναι η προάσπιση της ασφάλειας προσωπικού και φορτίου με αποφυγή συγκρούσεων μεταξύ πλοίων ή προσαράξεων.

Βάσει αυτής της πρόκλησης επιλέχθηκε η υλοποίηση μιας εφαρμογής πραγματικού χρόνου που θα ανιχνεύει περιπτώσεις εγγύτητας μεταξύ πλοίων, θα εξαγγέλει προειδοποιήσεις βάσει ορισμένων συνθηκών και θα οπτικοποιεί σε χάρτη περιοχές αυξημένης συχνότητας τέτοιων φαινομένων. Το αρχείο δεδομένων δεν είναι καθεαυτό πραγματικού χρόνου, αλλά μπορεί να μετασχηματιστεί σε ροή και να αποτελέσει το σύνολο εκπαίδευσης για την ανάπτυξη και έλεγχο του αλγόριθμου. Επίσης, τα δεδομένα ενδιαφέροντος ελήφθησαν από το σύστημα AIS που εξάγει διαρκώς realtime data, οπότε θα ήταν δυνατό να ενσωματωθεί με το σχεδιασμένο σύστημα μέσω κάποιου interface.

Σχετικές εργασίες & συστήματα

Το ζήτημα της ασφάλειας του Καναλιού της Μάγχης έχει αναλυθεί εκτενώς από διάφορους αρμόδιους φορείς (National Physical Laboratory, Royal Institute of Navigation, κά), αλλά δεν αποτελεί κεντρικό μέρος της παρούσας εργασίας ανάπτυξης συστήματος προληπτικής δράσης.

Το σύστημα που θα υλοποιηθεί βασίζεται πάνω σε δεδομένα από το σύστημα AIS, ο σχεδιασμός του οποίου έγινε εν μέρει λόγω της ανάγκης μείωσης συγκρούσεων μεταξύ πλοίων (Tetreault, Harati-Mokhtari, Hargreaves). Κάθε πλοίο με σύστημα AIS διαθέτει πομπό θέσης αλλά και δέκτη πληροφοριών για τις θέσεις και κινήσεις άλλων κοντινών πλοίων, τις οποίες λαμβάνει υπόψη ο πλοηγός κατά τη χάραξη πορείας. Πέρα από τους δέκτες σε πλοία υπάρχουν και επίγειοι δέκτες που ανήκουν σε οργανισμούς ή ιδιώτες οι οποίοι αποστέλλουν τις πληροφορίες σε συστήματα πληροφόρησης και παρατήρησης λιμενικών αρχών ή άλλων συστημάτων ναυσιπλοϊκού ενδιαφέροντος (πχ. MarineTraffic) όπου μπορούν να απεικονιστούν και να μελετηθούν από αρμόδιους ή μη. Συνεπώς, το ίδιο το πρωτόκολλο του AIS περιλαμβάνει το αρχικό τμήμα της εφαρμογής.

Η συγκεκριμένη εφαρμογή έχει το πλεονέκτημα της δυνατότητας κλιμάκωσης σε παγκόσμια κλίμακα (εν αντιθέση με τους περιορισμούς ενός AIS δέκτη πλοίου), αλλά και της ελεύθερης πρόσβασης μέσω ενός browser αντί εξειδικευμένου εξοπλισμού. Επιπλέον παρέχει εύκολη πρόσβαση σε πλατφόρμες επεξεργασίας και οπτικοποίησης για την εξαγωγή insights οποιασδήποτε φύσης (επιχειρηματικά, στατιστικά, ασφαλείας, κλπ).

Υλοποίηση

Ανάλυση δεδομένων και Big Data

Το σύστημα που θα σχεδιαστεί θα λαμβάνει ροή από συντεταγμένες πλοίων (μαζί με όποια βοηθητικά χαρακτηριστικά) ανά τακτικά χρονικά διαστήματα, θα αντιστοιχεί σε κάθε πλοίο δυο ακτίνες ασφαλείας (μικρή στατική/μεγάλη δυναμική), θα ελέγχει ανά πάσα στιγμή αν τέμνονται οι ακτίνες οποιωνδήποτε πλοίων και, σε αυτήν την περίπτωση, θα προειδοποιεί με κίτρινο ή με κόκκινο συναγερμό για πιθανή ή επικείμενη σύγκρουση.

Η υλοποίηση της ανάλυσης του dataset έγινε με μεθόδους και εργαλεία που χρησιμοποιούνται στην ανάλυση Μεγάλων Δεδομένων. Παρότι το μέγεθος και η συχνότητα ενημέρωσης του dataset δεν αντιστοιχούν στον ορισμό των Μεγάλων Δεδομένων και η ανάλυση θα μπορούσε να ολοκληρωθεί με κλασικές μεθόδους, λήφθηκε υπόψη η δυνατότητα επέκτασης του μοντέλου σε παγκόσμια κλίμακα. Καθώς το σύστημα AIS είναι παγκόσμιο, με κατάλληλη ροή από ειδικό tracker (πχ. MarineTraffic) είναι εφικτή η παρακολούθηση θέσεων εκατοντάδων χιλιάδων πλοίων με συχνότητα ενημέρωσης ανά λίγα δευτερόλεπτα, χωρίς να απαιτηθεί περαιτέρω κλιμάκωση του αλγόριθμου.

Το υπολογιστικό πλαίσιο που επιλέχθηκε ως βάση του συστήματος είναι το Apache Spark (Salloum, 2016). Οι λόγοι επιλογής είναι η ταχύτητα υπολογισμών συγκριτικά με άλλα κατανεμημένα υπολογιστικά πλαίσια, η ευκολία ανάπτυξης και λειτουργίας, και η συνδεσιμότητα με realtime dataflows μέσω Spark Streaming. Ο αλγόριθμος προγραμματίστηκε σε Python και εκτελείται στο Spark μέσω του PySpark interface. Αρχική έγινε υλοποίηση πρωτότυπου αλγόριθμου σε local single-node cluster (PC) με

σκοπό τη μεταφορά και τελική εκτέλεσή του σε multi-node cluster στο cloud (Okeanos). Θεωρητικά, κατά τη μεταφορά από local node σε cluster δε θα χρειαστούν σημαντικές μετατροπές στον αλγόριθμο.

Τα αποτελέσματα των υπολογισμών μπορούν να εισαχθούν με ή χωρίς προεργασία σε πλατφόρμες απεικόνισης χωροχρονικών δεδομένων και να απεικονιστούν τα σημεία ενδιαφέροντος σε χάρτη της περιοχής. Τέτοιες πλατφόρμες είναι μεταξύ άλλων τα Carto.com, NodeRed, Uber Kepler και ESRI ArcGIS,, που μπορούν να εξάγουν στατικά, δυναμικά ή και κινούμενα γραφήματα που βοηθούν στη συνολική κατανόηση προβλημάτων και οδηγούν στην εξαγωγή καλύτερων συμπερασμάτων.

Περιγραφή αρχιτεκτονικής αλγορίθμου

Η αρχιτεκτονική του αλγορίθμου αποτελείται από τα εξής μέρη:

1. **Stream** - Ανάγνωση αρχείων δεδομένων και εξαγωγή σε ροή δεδομένων
2. **Input** μεταβαλλόμενων (θέση και κίνηση πλοίων) και στατικών δεδομένων (τεχνικά χαρακτηριστικά πλοίων, λιμένες, κλπ)
3. **Join** μεταβαλλόμενων/στατικών δεδομένων και δημιουργία Minimum Bounding Rectangle, Grid Partitioning, KD-Tree λιμένων
4. **Map** - Υπολογισμός ακτίνων προειδοποίησης για κάθε πλοίο και δημιουργία <cell,ship> tuples
5. **Window** - Φιλτράρισμα τελευταίου στίγματος πλοίων εκτός λιμένων για τουλάχιστον 3 λεπτά
6. **Reduce** - Σύγκριση αποστάσεων μεταξύ στιγμάτων πλοίων εντός κάθε cell
7. **Output** προειδοποιήσεων βάσει απόστασης πλοίων σε αρχείο ή stream
8. **Visualization** αποτελεσμάτων

Εκτέλεση σε single node

Για την υλοποίηση του πρωτότυπου αλγορίθμου βάσει της προηγούμενης αρχιτεκτονικής σε PySpark δημιουργήθηκαν δυο scripts, ένα για τη δημιουργία του data stream και ένα για τον υπολογισμό και εξαγωγή των προειδοποιήσεων σε Spark Streaming.

Stream, Input & Join (1, 2, 3)

Τα σύνολα δεδομένων εισόδου χωρίζονται σε στατικά και μεταβαλλόμενα. Οι υπολογισμοί που θα γίνουν περιλαμβάνουν και τα δυο είδη και το Spark επιτρέπει τη φόρτωση των στατικών με απλή μέθοδο read file από το τοπικό filesystem, αλλά η είσοδος των μεταβαλλόμενων γίνεται μέσω data stream. Το Spark Stream υποστηρίζει είσοδο ροής από Kafka, Flume, Twitter, μεταξύ άλλων, αλλά και από το HDFS/local FS αναζητώντας realtime μεταβολές σε αρχεία εντός κάποιου προσδιορισμένου φακέλου. Στην τοπική υλοποίηση επιλέχθηκε η είσοδος του stream από το τοπικό FS καθώς είναι το απλούστερο και ταχύτερο σε υλοποίηση, χωρίς να απαιτεί ρυθμίσεις πολύπλοκων παραμέτρων.

Το sparkStream script καλείται σε python console και εκτελεί ανάγνωση των αρχείων στιγμάτων (μεταβαλλόμενα) και των χαρακτηριστικών πλοίων (στατικά), εκτελεί ένωσή τους με κλειδί τη στήλη SourceMMSI των αρχείων και απορρίπτει λανθασμένες ή κενές εγγραφές. Τέλος, δημιουργεί προσομοίωση ροής εξάγοντας χρονολογικά τις εγγραφές βάσει του timestamp, εισάγοντας καθυστέρηση όσο και οι χρονικές διαφορές μεταξύ τους. Υπάρχει δυνατότητα επιτάχυνσης της ροής από πραγματικό χρόνο έως σχεδόν άμεσο (περιορισμένο από την ταχύτητα ανάγνωσης του FS). Η έξοδος της ροής δημιουργεί αρχεία ανά timestamp απευθείας εντός του φακέλου ανάγνωσης του Spark Stream.

Το δεύτερο script που υπολογίζει και εξάγει τις προειδοποιήσεις καλείται στο Spark αφότου έχει ξεκινήσει η ροή. Μετά την αρχικοποίηση του συστήματος Spark ενεργοποιείται ένας listener (textFileStream) που παρατηρεί το φάκελο που δίνεται ως όρισμα της εντολής εκτέλεσης για είσοδο νέων αρχείων ή μεταβολές σε αυτά. Το δεύτερο όρισμα T/F μεγιστοποιεί τη ροή για δοκιμές του αλγορίθμου.

Κατά την κλήση του το script αρχικοποιεί ένα Minimum Bounding Rectangle και ένα KD-Tree λιμένων. Το MBR είναι ένα ορθογώνιο σύνορο που περιορίζει το χώρο υπολογισμών στο ελάχιστο απαραίτητο, στη συγκεκριμένη περίπτωση [N:51°, S:45°, E:0°, W:-10°]. Έπειτα, για βελτίωση απόδοσης μέσω αποφυγής αχρείαστων συγκρίσεων μεταξύ πολύ απομακρυσμένων πλοίων αλλά και βελτιωμένη παραλληλοποίηση με σχηματισμό tuples, το MBR χωρίζεται εξολοκλήρου σε grid με κελιά μεγέθους 1°x1°. Τέλος, για την μετέπειτα ταχύτερη ανάγνωση των συντεταγμένων λιμένων, οι συντεταγμένες διαβάζονται άπαξ από το αρχείο shape

μέσω του Fiona library και τοποθετούνται σε KD-Tree καθώς είναι από τις ταχύτερες σε προσπέλαση δομές στατικών δεδομένων.

Map & Window (4, 5)

Τα αρχεία ροής καλύπτουν χρονική διάρκεια ενός δευτερολέπτου έκαστο, περιλαμβάνουν τις εγγραφές όλων των στιγμάτων πλοίων που εκπέμφθηκαν σε εκείνο το timestamp και εισάγονται στο Spark ως δομή εισόδου RDD διάρκειας 1-5 δευτερολέπτου. Οι εγγραφές του RDD χωρίζονται ανά πλοίο, εισάγονται αντιπροσωπευτικά ως στίγμα στο πλαίσιο του Spark, οργανώνονται σε διάνυσμα [ID,speed,lon,lat,time,callsign,length] με τη μέθοδο parse_rdd, τους προστίθενται οι υπολογισμένες ακτίνες προειδοποίησης(μικρή συναρτήσει στατικού μήκους και μεγάλη συναρτήσει μεταβαλλόμενης ταχύτητας), και με τη μέθοδο grid_locate αντιστοιχίζεται κάθε στίγμα πλοίου με το κελί στο οποίο ανήκει και πιθανά γειτονικά αυτού. Η μέθοδος εκτελεί Mapping επιστρέφοντας tuples μορφής <cell,ship> όπου key είναι κελιά στα οποία αντιστοιχίζεται το στίγμα και value είναι το διάνυσμα του στίγματος με όλα τα χαρακτηριστικά του πλοίου, συμπεριλαμβανομένων των ακτίνων προειδοποίησης. Σκοπός αυτής της σύνθεσης των tuples είναι σε μετέπειτα βήμα να γίνει Reduce συγκρίνοντας στίγματα πλοίων που βρίσκονται πράγματι σε συγκρίσιμες αποστάσεις μεταξύ τους.

Σε αυτό το σημείο εκτελείται μια προεπεξεργασία των δεδομένων εισόδου. Κάποιες από τις παρακάτω διεργασίες θα μπορούσαν ενδεχομένως να υλοποιηθούν σε πρότερα στάδια (επίπεδο αρχείου εισόδου, ροής ή διανύσματος αντί σε tuples) με σκοπό την αποφυγή περιττών μετασχηματισμών, αλλά καθώς δεν είναι αυτός ο άμεσος στόχος επιλέχθηκε αυτό το σημείο ως λογική θέση στην αρχιτεκτονική του πρωτότυπου κώδικα.

Αρχικά, καθώς δε στέλνουν όλα τα πλοία το AIS στίγμα τους την ίδια στιγμή ή με την ίδια περίοδο, απαιτείται να δημιουργηθεί ένα κυλιόμενο παράθυρο διάρκειας τουλάχιστον μεγαλύτερης από τη μεγαλύτερη περίοδο. Αυτή είναι διάρκειας περίπου 3 λεπτών για πλοία που βρίσκονται σε αγκυροβόλι, συνεπώς το απαιτούμενο ελάχιστο κινούμενο παράθυρο RDD ορίστηκε στα 200 δευτερόλεπτα. Περίοδος ανανέωσης του παραθύρου ορίστηκε στα 2-10 δευτερόλεπτα καθώς δεν μπορεί να γίνει ίση ή μικρότερη της διάρκειας του RDD.

Επόμενο στάδιο είναι η υλοποίηση φίλτρου που απομακρύνει tuples πλοίων εντός ή πλησίον λιμένων, καθώς πρόκειται για περιοχές που πλοία βρίσκονται συχνά κοντά σε άλλα σε ασφαλείς συνθήκες. Αυτό επιτυγχάνεται υπολογίζοντας την απόσταση Haversine μεταξύ κάθε πλοίου και όλων των λιμένων του αρχείου εισόδου περασμένα σε KD-Tree για γρήγορη αναζήτηση. Η μέθοδος αναζήτησης των φύλλων/λιμένων είναι μέρος της αντίστοιχης βιβλιοθήκης. Πλοία που απέχουν λιγότερο από 2χμ από τις συντεταγμένες του αρχείου λιμένων διαγράφονται από τη ροή. Θα ήταν εφικτή περαιτέρω βελτιστοποίηση σε αυτό το σημείο χωρίζοντας το αρχικό KD-Tree σε subtrees ανάλογα με το gridcell στο οποίο ανήκουν για αποφυγή περιττών υπολογισμών, αλλά μια τέτοια υλοποίηση είναι μάλλον υπερβολική σε αυτό το σημείο.

Τέλος, σε αυτό το σημείο κάθε κελί περιέχει tuples που αντιστοιχούν σε στίγματα του ίδιου πλοίου σε διαφορετικές χρονικές στιγμές εντός των τελευταίων 200 δευτερολέπτων. Με τη μέθοδο keep_latest διατηρούνται μόνο το πλέον πρόσφατα στίγματα κάθε πλοίου σε κάθε κελί. Με αυτόν τον τρόπο θεωρείται ότι αν ένα πλοίο παύσει να εκπέμπει για 3 δευτερόλεπτα αυτό βρίσκεται ακίνητο στο σημείο του τελευταίου στίγματος για 3 λεπτά. Σε τελική υλοποίηση θα πρέπει να υλοποιηθεί αλγόριθμος που θα ελέγχει την κατάσταση του πλοίου (πχ. εν πλω, αγκυροβόλι, άλλο) και τη θέση του σχετικά με συνοριακά κελιά (αν διέσχισε σύνορο κελιού) ώστε να κρίνει πόσο χρονικό διάστημα θα πρέπει να παραμείνει το τελευταίο στίγμα στη μνήμη, ή αν έγινε κάποιο απρόβλεπτο γεγονός που χρήζει επέμβασης (πχ. βλάβη, βύθιση, άλλο).

Reduce, Output & Visualization (6, 7, 8)

Στο τελευταίο τμήμα ολοκληρώνεται η υλοποίηση του MapReduce, αν και όχι με ρητή συνάρτηση Reduce, αλλά με προσέγγισή της μέσω συνδυασμού άλλων ειδών συναρτήσεων (groupByKey & mapValues). Τα tuples (τελευταία στίγματα πλοίων) ομαδοποιούνται κατά το κελί τους και γίνονται συγκρίσεις αποστάσεων βάσει συνάρτησης Haversine μεταξύ όλων των πλοίων του ίδιου κελιού. Ζεύγη πλοίων που βρίσκονται σε απόσταση μικρότερη του αθροίσματος των μεγάλων ακτίνων τους επιστρέφονται με κίτρινο συναγερμό, ενώ τα ζεύγη σε απόσταση μικρότερη των μικρών ακτίνων επιστρέφονται με κόκκινο συναγερμό.

Τα ζεύγη των αποτελεσμάτων εξάγονται διαδοχικά σε αρχείο CSV. Αυτά φορτώνονται με ελαφρά επεξεργασία (τυποποίηση σε template) στην online πλατφόρμα του Carto για απεικόνιση των σημείων σε θαλάσσιο χάρτη, επαλήθευση των αποτελεσμάτων, και εντοπισμό πιθανών περιοχών αυξημένης επικινδυνότητας. Ακόμα, είναι δυνατή η απεικόνιση των λιμένων, heatmaps των σημείων προειδοποίησης, τα χαρακτηριστικά των πλοίων και οι ακτίνες τους, αλλά και οι κινούμενες τροχιές των πλοίων σε πραγματικό ή μη χρόνο.

Εκτέλεση σε cluster

Η μεταφορά ενός αλγόριθμου που τρέχει ορθά από τοπικό Spark σε Spark cluster θεωρητικά αναμένεται να είναι μια στρωτή διαδικασία χωρίς ανάγκη σημαντικών τροποποιήσεων στον κώδικα. Ωστόσο, κατά τη μεταφορά του πρωτότυπου κώδικα στο cluster ανέκυψε πλήθος ζητημάτων που απαίτησαν διεξοδικές μετατροπές στις μεθόδους εκτέλεσης, αλλά όχι στην ίδια αρχιτεκτονική και λογική του αλγορίθμου. Τα ζητήματα αυτά σε μεγάλο βαθμό οφείλονταν σε διαφορές λειτουργίας μεταξύ HDFS και local FS, αλλά και σε έλλειψη οικειότητας με μεθόδους βελτιστοποίησης κώδικα Python. Τέλος, παρατηρήθηκαν περιπτώσεις όπου λόγω παραλείψεων ο πρωτότυπος κώδικας απέκρυπτε επιθυμητά αποτελέσματα και έγινε προσπάθεια να περιοριστούν με το νέο κώδικα του cluster.

Δομή cluster

Το Spark εγκαταστάθηκε σε cluster στον Okeano με 5 nodes (1 master & 4 slaves). Τα nodes διαθέτουν έκαστο 4πύρηνες CPU, 8GB RAM και 60 GB RAM. Τρέχουν Ubuntu Server LTS στο οποίο εγκαταστάθηκε Hadoop με YARN, και πάνω σε αυτό εγκαταστάθηκαν τα Spark, Kafka, pySpark και οι απαραίτητες Python libraries. Δοκιμάστηκε η λειτουργία τρέχοντας το WordCount demo και περάστηκαν τα απαραίτητα αρχεία δεδομένων και κώδικα σε φακέλους του HDFS.

Stream, Input & Join (1, 2, 3)

Η βασικότερη διαφορά που αντιμετωπίστηκε κατά την εκτέλεση του αλγορίθμου στο HDFS συγκριτικά με το local node είναι η αδυναμία μεταφοράς της μεθόδου δημιουργίας του stream. Στο local FS τα αρχεία του stream γράφονται απευθείας στο φάκελο ανάγνωσης της ροής ενώ στο HDFS για να υλοποιηθεί η ίδια λειτουργία θα

χρειάζονταν πολλαπλές μετακινήσεις αρχείων (λόγω αδυναμίας ταυτόχρονης προσπέλασης ανάγνωσης/εγγραφής), προσθέτοντας στο χρόνο προσπέλασης των αρχείων. Εξαιτίας αυτού του προβλήματος αποφασίστηκε η χρήση messaging server Kafka για τη δημιουργία ροών που υποστηρίζει και αναγνωρίζει το Spark. Kafka χρησιμοποιήθηκε και στη ροή εξόδου των αποτελεσμάτων. Ο Kafka server κάνει ανάγνωση των αρχείων των πλοίων, ελέγχει για ορθότητα τα δεδομένα, κάνει join των σχετικών attributes και εξάγει σε DStream τα δεδομένα των πλοίων βάσει του timestamp με την αντίστοιχη καθυστέρηση.

Το DStream του Kafka εκπέμπεται από ένα producer και λαμβάνεται από το Spark μέσω του port 9092. Μετά την είσοδο στο Spark τα δεδομένα επεξεργάζονται όπως στον πρωτότυπο αλγόριθμο. Παράλληλα γίνεται αρχικοποίηση του Grid Partitioning και του KD-Tree.

Map, Window & Reduce (4, 5, 6)

Στις διεργασίες αυτές δεν έγιναν αλλαγές αρχιτεκτονικής αλλά βελτιστοποίησης για ταχύτερη επεξεργασία της ροής και περιορισμό απωλειών true positives λόγω σφαλμάτων στον πρωτότυπο κώδικα.

Output & Visualization (7, 8)

Η έξοδος των αποτελεσμάτων του Spark δε γίνεται απευθείας σε αρχείο CSV όπως με τον πρωτότυπο κώδικα, αλλά συνδέεται με έναν Kafka consumer που λαμβάνει τα διανύσματα εξόδου από το Spark στο Port 9092 και τα προσθέτει σειριακά σε ένα CSV αρχείο στο local FS του masternode[2]. Αυτό έγινε για να είναι εφικτή η πρόσβαση στο αρχείο από απομακρυσμένες IP στο Διαδίκτυο μέσω HTTP server. Εάν το Spark έγραφε απευθείας σε αρχείο CSV αυτό θα βρισκόταν στο HDFS και, ή θα ήταν απαραίτητη η εκτέλεση shell script για να αντιγράφεται στο local FS σε τακτά διαστήματα, ή οι απομακρυσμένες IP θα χρειάζονταν credentials για τη σύνδεση στο HDFS του cluster. Επίσης διατίθεται αρχείο εκτέλεσης 6 ωρών ως demo[3].

Μέσω του HTTP server είναι εφικτή η πρόσβαση online πλατφόρμων ανάλυσης δεδομένων στο αρχείο των αποτελεσμάτων σε πραγματικό χρόνο, ενώ ταυτόχρονα το Spark Stream μπορεί να βρίσκεται ακόμα σε λειτουργία και να εξάγει πρόσθετα

[2] LIVE αρχείο αποτελεσμάτων: http://83.212.100.182:443/alerts.csv
[3] DEMO αρχείο αποτελεσμάτων: http://83.212.100.182:443/alerts_6hours.csv

αποτελέσματα. Η ανάλυση που έγινε στα πλαίσια της εργασίας περιορίστηκε στην οπτικοποίηση των προειδοποιήσεων σε χάρτη με τη βοήθεια δυο online πλατφόρμων, του Carto.com και του NodeRed.

Εκτέλεση αλγορίθμου

Η εκτέλεση του αλγορίθμου γίνεται με τη διαδοχική κλήση τεσσάρων shell-scripts στο terminal του cluster:

1. Step-1-streaming-consumer.sh: Αρχικοποιεί το Kafka εξόδου για να λάβει τη ροή του Spark (έξοδος προειδοποιήσεων)
2. Step-2-spark-submit.sh: Αρχικοποιεί το Spark Streaming για να λάβει τη ροή του Kafka εισόδου (έξοδος στιγμάτων πλοίων)
3. Step-3-streaming-producer.sh: Καλεί το Kafka εισόδου να διαβάσει το αρχείο εισόδου και να εξάγει τη ροή στιγμάτων
4. Step-4-http-server.sh: Ενεργοποιεί τον HTTP server για την πρόσβαση στο αρχείο εξόδου μέσω IP στο port 443

Αποτελέσματα

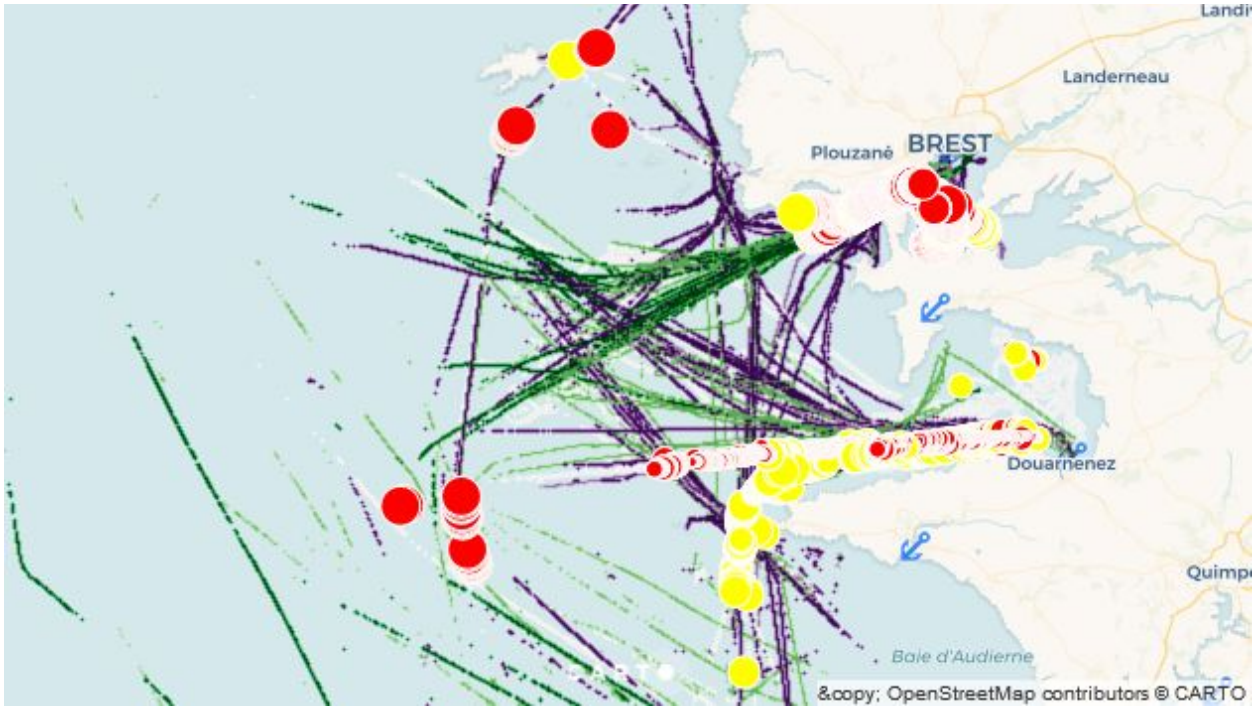
Οπτικοποίηση σε realtime πλατφόρμες

Carto.com

Το Carto.com είναι μια υπολογιστική πλατφόρμα SaaS στο cloud που παρέχει GIS και εργαλεία χαρτογράφησης για οπτικοποίηση δεδομένων μέσα σε web browser. Επιτρέπει την πρόσβαση σε πολλά είδη αρχείων με χωροχρονικές πληροφορίες, είτε αυτά βρίσκονται τοπικά, στο cloud ή σε network clusters. Για τις ανάγκες της εργασίας έγινε εισαγωγή και απεικόνιση τριων συνόλων δεδομένων: λιμένες από τοπικό αρχείο, δυναμικές τροχιές πλοίων από τοπικό αρχείο, και προειδοποιήσεις συγκρούσεων πλοίων από το spark cluster με πρόσβαση πραγματικού χρόνου.

Οι λιμένες χαρτογραφήθηκαν βάσει των συντεταγμένων τους και απεικονίζονται με γαλάζια άγκυρα. Οι τροχιές πλοίων μιας εβδομάδας φορτώθηκαν από το αρχικό αρχείο AIS, αντιστοιχήθηκαν με διαφορετικό χρώμα ανά MMSI πλοίου, και απεικονίστηκαν με animation βάσει του timestamp που φέρουν. Τέλος, στο πρότζεκτ

φορτώθηκε το αρχείο προειδοποιήσεων από το spark cluster με ενημέρωση ανά μια ώρα. Οι συντεταγμένες κάθε προειδοποίησης λήφθηκαν ως η πιο πρόσφατη θέση εκ του ζεύγους πλοίων της, το σημείο χρωματίστηκε βάσει του επιπέδου συναγερμού (κίτρινο/κόκκινο) και το μέγεθός του καθορίστηκε από την ελάχιστη απόσταση μεταξύ των πλοίων. Είναι δυνατή και η απεικόνιση επιπλέον πληροφοριών (ζεύγος πλοίων, ώρα, κά) αλλά κάτι τέτοιο θα είχε επίπτωση στην αναγνωσιμότητα του χάρτη. Παρατίθεται στιγμιότυπο απεικόνισης και σύνδεσμος για τον online χάρτη[4]:



NodeRed

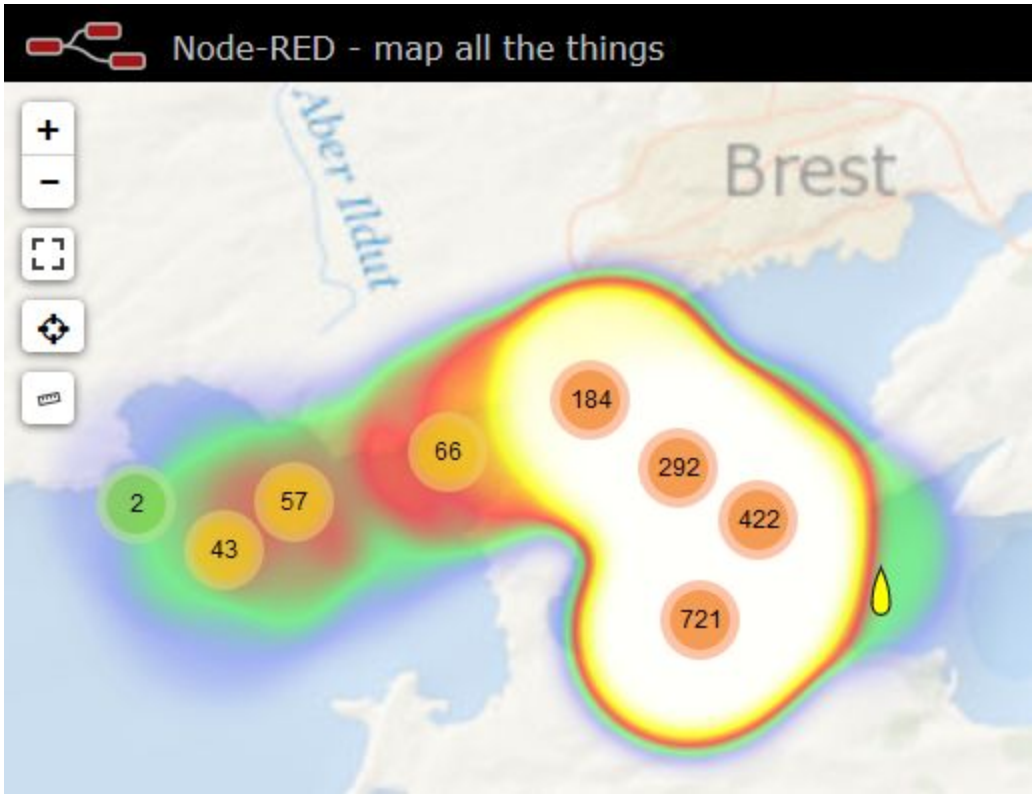
Το NodeRed είναι ένα flow-based εργαλείο ανάπτυξης εφαρμογών για τη διασύνδεση συσκευών, APIs και online υπηρεσιών ως τμήμα του Internet of Things. Παρέχει ένα browser-based flow editor που επιτρέπει την εύκολη σύνδεση λειτουργιών (nodes) για τη μεταφορά, επεξεργασία και ανάλυση μηνυμάτων πληροφορίας. Το NodeRed είναι ένα ελαφρύ runtime βασισμένο στο Node.js και είναι ιδανικό software για εκτέλεση σε edge-of-network εφαρμογές και υλικό χαμηλού κόστους (πχ. Raspberry Pi), καθώς και στο cloud.

Η υλοποίηση realtime εφαρμογής χαρτογράφησης προειδοποιήσεων με NodeRed είναι απλή διαδικασία τοποθέτησης συγκεκριμένων nodes σε σειρά:

[4] http://goo.gl/zwYR7y

1. Interval Trigger - επανάληψη αλγορίθμου ανά 1-10 sec
2. HTTP Request - ανάκτηση αρχείου αποτελεσμάτων μέσω IP
3. CSV - μετατροπή σε JS object/message flow
4. Function - τυποποίηση του message payload για χαρτογράφηση
5. Map/Output - απεικόνιση σημείων βάσει συντεταγμένων και επιπέδου συναγερμού στο χάρτη και στο terminal

Εναλλακτικά, αντί για ανάκτηση ολόκληρου του CSV θα μπορούσε να γίνει σύνδεση του NodeRed μέσω MQTT node απευθείας με την έξοδο του Kafka. Αυτό θα επέτρεπε άμεση ενημέρωση του χάρτη και πληρέστερη διαχείριση του χρονικού παραθύρου παρακολούθησης. Παρατίθεται χάρτης με ενδεικτικές θέσεις συναγερμών και realtime heatmap θαλάσσιων περιοχών με μεγαλύτερη συχνότητα αυξημένης εγγύτητας πλοίων.



Συμπεράσματα

Αποτελέσματα οπτικοποίησης

Καθώς το αντικείμενο της εργασίας ήταν η υλοποίηση ενός συστήματος επεξεργασίας και ανάλυσης σε πραγματικό χρόνο, δόθηκε περιορισμένη προσοχή στην εξαγωγή συμπερασμάτων βάσει ολόκληρου του συνόλου δεδομένων. Το μεγαλύτερο μέρος των αποτελεσμάτων είναι περιορισμένο στο λιμάνι της Brest και, καθότι είναι εφαρμογή πραγματικού χρόνου, οι δοκιμές περιορίστηκαν σε διάρκεια max 24 ωρών. Συνεπώς δεν είναι εφικτή η εξαγωγή χρήσιμων συμπερασμάτων.

Ωστόσο, από την πολύ περιορισμένη δυνατότητα απεικόνισης με το Carto και το NodeRed παρατηρήθηκε, όπως αναμενόταν άλλωστε, ότι τα περισσότερα γεγονότα έλαβαν χώρα μόλις εκτός του λιμένα της Brest και σε περιοχές εκτός κυρίων θαλάσσιων διαδρομών εμπορίου, δηλαδή περιοχές αλιείας. Επίσης παρατηρήθηκε ότι στις περισσότερες περιπτώσεις οι προειδοποιήσεις για ένα ζεύγος πλοίων επαναλαμβάνονταν τακτικά και εν πλω, συνεπώς δεν επρόκειτο για περιπτώσεις συγκρούσεων αλλά παράλληλης πλεύσης. Τέλος, εμφανίστηκαν ορισμένα πολύ σπάνια γεγονότα κίτρινου συναγερμού στα ανοιχτά χωρίς επανάληψη, το οποίο δείχνει επιτυχή αποφυγή σε ασφαλή απόσταση.

Περιορισμοί συστήματος

Από τη λειτουργία και τις δοκιμές που έγιναν πάνω στο σύστημα προέκυψαν τα ακόλουθα συμπεράσματα.

Η ταχύτητα ροής συναρτήσει των απαιτήσεων στην επεξεργασία έχουν επίπτωση στην ακρίβεια του τελικού αποτελέσματος. Σε περίπτωση που το YARN δεν προλάβει να εκτελέσει συγκεκριμένους υπολογισμούς μένει πίσω στο χρόνο προκειμένου να τους ολοκληρώσει, με αποτέλεσμα να φορτώνονται νέα RDDs στη μνήμη και, μόλις αυτή γεμίσει, να χάνονται αποτελέσματα. Για τη βέλτιστη απόδοση του αλγορίθμου απαιτούνται περαιτέρω δοκιμές με διάφορες ταχύτητες ροής, με αλλαγές θέσης των μεθόδων φιλτραρίσματος εγγραφών, με τροποποιήσεις στο μέγεθος RDD, window και update rate, και τέλος, με επιπλέον nodes στο cluster για καλύτερο παραλληλισμό.

Τέλος, η ανάκτηση των αποτελεσμάτων από τις πλατφόρμες γίνεται με GET ολόκληρου του αρχείου αποτελεσμάτων κάθε φορά, αντί με PUSH μόνο των πλέον πρόσφατων αποτελεσμάτων από το Kafka, με αποτέλεσμα να μην επιτυγχάνεται το βέλτιστο update rate.

Επίλογος

Στην εργασία αυτή υλοποιήθηκε σύστημα πραγματικού χρόνου για την αναζήτηση συμβάντων υπερβολικής εγγύτητας δυο πλοίων και οπτικοποίησης θέσεων ενδιαφέροντος σε χάρτες. Το δοσμένο σύνολο δεδομένων αντιμετωπίστηκε ως ροή βάσει χρονοσειράς και επεξεργάστηκε σε υπολογιστικό πλαίσιο Μεγάλων Δεδομένων. Αρχικά σχεδιάστηκε πρωτότυπος αλγόριθμος σε local node, ο οποίος μεταφέρθηκε με μετατροπές σε υπολογιστικό spark cluster. Η οπτικοποίηση των αποτελεσμάτων έγινε με τη βοήθεια online πλατφόρμων που επιτρέπουν την απεικόνιση δυναμικών δεδομένων πραγματικού χρόνου και την επεξεργασία τους με σκοπό την εξαγωγή επιπλέον insights.

Κατά το σχεδιασμό έγινε εξοικείωση με τεχνολογίες Μεγάλων Δεδομένων, τις απαιτήσεις και τις τεχνικές δυσκολίες που μπορούν να εμφανιστούν κατά την πρακτική εφαρμογή τους. Από τα ίδια τα δεδομένα δεν ήταν εφικτή η εξαγωγή κάποιων στέρεων συμπερασμάτων καθώς δεν ήταν αυτό το θέμα ανάπτυξης. Για την εξαγωγή συμπερασμάτων θα ήταν απαραίτητη η χρήση άλλης τεχνολογίας (Hadoop, Spark, κλπ) και αρχιτεκτονικής (block file vs stream) ώστε να διευκολυνθεί η διαδικασία της εξόρυξης και μηχανικής μάθησης.

Αναφορές

RAY, Cyril, DRÉO, Richard, CAMOSSI, Elena, & JOUSSELME, Anne-Laure. (2018). Heterogeneous Integrated Dataset for Maritime Intelligence, Surveillance, and Reconnaissance (Version 0.1) [Data set]. Zenodo. http://doi.org/10.5281/zenodo.1167595

Shamseer Mambra (2017), The Strait Of Dover – The Busiest Shipping Route In The World

Mambra , Shamseer (2017, August 1). The Strait Of Dover – The Busiest Shipping Route In The World. Retrieved from https://www.marineinsight.com/marine-navigation/the-strait-of-dover-the-busiest-shipping-route-in-the-world/

B. J. Tetreault, "Use of the Automatic Identification System (AIS) for maritime domain awareness (MDA)," *Proceedings of OCEANS 2005 MTS/IEEE*, Washington, DC, 2005, pp. 1590-1594 Vol. 2.

Harati-Mokhtari, A., Wall, A., Brooks, P., & Wang, J. (2007). Automatic Identification System (AIS): Data Reliability and Human Error Implications. *Journal of Navigation, 60*(3), 373-389. doi:10.1017/S0373463307004298

Hargreaves, E. (1973). Safety of Navigation in the English Channel. *Journal of Navigation, 26*(4), 399-407. doi:10.1017/S0373463300021524

Salloum, S., Dautov, R., Chen, X. et al. Int J Data Sci Anal (2016) 1: 145. https://doi.org/10.1007/s41060-016-0027-9