****

**ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ**

**ΠΜΣ ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΥΠΗΡΕΣΙΕΣ**

**Μεγάλα Δεδομένα και Αναλυτική II: Τεχνικές και Εργαλεία**

****

**Χρήστος Φιλιππής ME1737**

**ΑΘΗΝΑ ΣΕΠΤΕΜΒΡΙΟΣ 2018**

## Πίνακας περιεχομένων

[**Πίνακας περιεχομένων**](#_l5ijjiifqjv9) **2**

[**Εισαγωγή**](#_yzj8s5hpt8se) **3**

[**Τεχνική υλοποίηση**](#_tuqu4schk10j) **3**

[Αλγόριθμος](#_6xsg4giqgqex) 3

[Γλώσσα προγραμματισμού και εργαλεία](#_z6x9i3fyt1rh) 3

[Πηγαίος κώδικας](#_ueishzoi6ic8) 3

[Δεδομένα](#_gqk5ahb6dhku) 4

[Υποδομή](#_am2788fpxgnh) 4

[**Εκτέλεση προγράμματος**](#_xoukvi214pa8) **4**

[Εισαγωγή δεδομένων στο Hdfs](#_xz740080u8ed) 4

[Spark submit](#_7u9fosilp7h4) 5

[Το πρόγραμμα](#_nvixfom59adt) 6

[Οπτικοποίηση Δεδομένων](#_jt7p93tlza6) 7

[Χρόνος εκτέλεσης](#_e4cnkwp6gpzm) 7

[**Αποτελέσματα**](#_xwztg1mspcjt) **7**

[**Βιβλιογραφία**](#_r31tx7mm3gu9) **9**

## Εισαγωγή

Στην εργασία αυτή καλούμαστε να υλοποιήσουμε ένα πρόγραμμα το οποίο μπορεί να εντοπίσει περιοχές οι οποίες έχουν χρησιμοποιούνται περισσότερο για αλιεία.

Η βασική ιδέα να ήταν να μπορούμε να αναλύσουμε τροχιές από αλιευτικά πλοία προκειμένου να εντοπίσουμε περιοχές με αυξημένη πυκνότητα εμφάνισης αυτών των πλοίων. Για την εύρεση αυτών των περιοχών θα χρησιμοποιήσουμε το αλγόριθμο BigCAB, και μέσω της εφαρμογής θα οπτικοποιήσουμε περιοχές (2D κελιά) με ασυνήθιστη υψηλή πυκνότητα με τρόπο όμοιο με αυτό τον heat maps, ώστε να μπορεί κάποιος άνθρωπος να καταλάβει εύκολα ποιες περιοχές υποδεικνύει το σύστημα και να τις ερμηνεύσει.

## Τεχνική υλοποίηση

### Αλγόριθμος

Σε πρώτη φάση βασιστήκαμε στο paper της ερευνητικής ομάδας του Πανεπιστημίου Πειραιώς (https://www.ds.unipi.gr/prof/cdoulk/papers/gis16.pdf) και στα συμπεράσματα στα οποία καταλήγει αυτή η μελέτη, προκειμένου να προσεγγίσουμε τη λύση του προβλήματος όσον αφορά κυρίως τον αλγόριθμο αλλά και τα εργαλεία που χρησιμοποιούνται. Οπότε η εφαρμογή ξεκινώντας φορτώνει τα δύο σύνολα δεδομένων, στην συνέχεια γίνεται μια σύνδεση (left outer join) στα παραπάνω σύνολα χρησιμοποιώντας το αναγνωριστικό του κάθε πλοίου προκειμένου να εντοπιστούν ποια πλοία είναι αλιευτικά.

Αφού πλέον έχουμε τροχιές από αλιευτικά πλοία πλέον μπορούμε να προχωρήσουμε στην ανάλυση θερμών περιοχών με μεγάλη πυκνότητα εμφάνισης τέτοιων πλοίων.

Αφου χωρίσουμε το διαθέσιμο χώρο σε κουτάκια συγκεκριμένων διαστάσεων και βρουμε το βάρος που έχει το κάθε κουτάκι σύμφωνα με τον Getis-ord εντοπίζουμε τα πιο θερμά σημεία. Στην συνέχεια οι περιοχές αυτές εμφανίζονται σε έναν διαδραστικό χάρτη στον οποίο ο χρήστης μπορεί να πλοηγηθεί χρησιμοποιώντας έναν browser.

### Γλώσσα προγραμματισμού και εργαλεία

Στην υλοποίηση αυτή έχει επιλεγεί η γλώσσα Python 2.7 καθώς είναι μια εύκολη, εύχρηστη και πολύ ισχυρή γλώσσα. Για την γρήγορη εκτέλεση του προγράμματος χρησιμοποιείται Spark 2.2 και για την οπτικοποίηση των αποτελεσμάτων σε χάρτη έχει χρησιμοποιηθεί το module gmplot.

### Πηγαίος κώδικας

<https://github.com/chrisfilippis/fishing_hotspots>

### Δεδομένα

Τα δεδομένα που χρησιμοποιούμε για τις τροχιές των πλοίων είναι δεδομένα που έχουν συλλεχθεί από το AIS της Ναυτικής ακαδημίας της Γαλλίας (Naval Academy, France, ecole-navale.fr).

<https://zenodo.org/record/1167595>

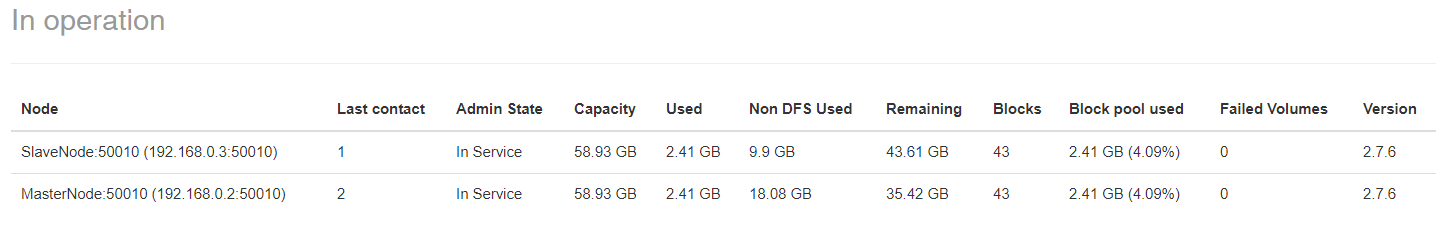
Το δεύτερο σύνολο δεδομένων που χρησιμοποιείται είναι η λίστα με όλα τα αλιευτικά πλοία εγγεγραμμένα στη ANFR (French Frequencies agency).

<https://www.data.gouv.fr/fr/datasets/donnees-radio-maritime>

Τα σύνολα δεδομένων αυτά έχουν ένα κοινό πεδίο το οποίο μας επιτρέπει την συσχέτιση τους. Το πεδίο αυτό (mmsi, Maritime Mobile Service Identity) είναι το αναγνωριστικό του κάθε πλοίου.

### Υποδομή

Για την εκτέλεση του προγράμματος χρησιμοποιήθηκε Spark 2.2 το οποίο αποτελείται από ένα cluster δύο κόμβων. Στο ίδιο cluster έχει στηθεί και ένα Hadoop Yarn προκειμένου να το χρησιμοποιήσουμε για να αποθηκεύουμε τα δεδομένα κατανεμημένα. Οπότε αυτό το κατανεμημένο χρησιμοποιεί τους πόρους δύο κόμβων προκειμένου να εκτελέσει παράλληλα διεργασίες αλλά και για την αποθήκευση των δεδομένων. Στο cluster αυτό υπάρχει ένας master κόμβος και ένας slave. Ο master έχει πρόσβαση στο internet σε αντίθεση με τον slave.



Τα hadoop nodes του hdfs

## Εκτέλεση προγράμματος

### Εισαγωγή δεδομένων στο Hdfs

Το πρώτο βήμα για την εκτέλεση του προγράμματος ήταν να ανεβάσουμε τα δεδομένα στο yarn. Τα δεδομένα μας αποτελούν δύο αρχεία, έναν για κάθε σύνολο οπότε ανεβαίνουν στο yarn για να είναι διαθέσιμα προς όλους τους κόμβους. Οπότε σε πρώτο βήμα κατεβάσαμε τα αρχεία που μας ενδιέφεραν και στην συνέχεια τα ανεβάσαμε στο Hdfs με τις παρακάτω εντολές αναλόγως.

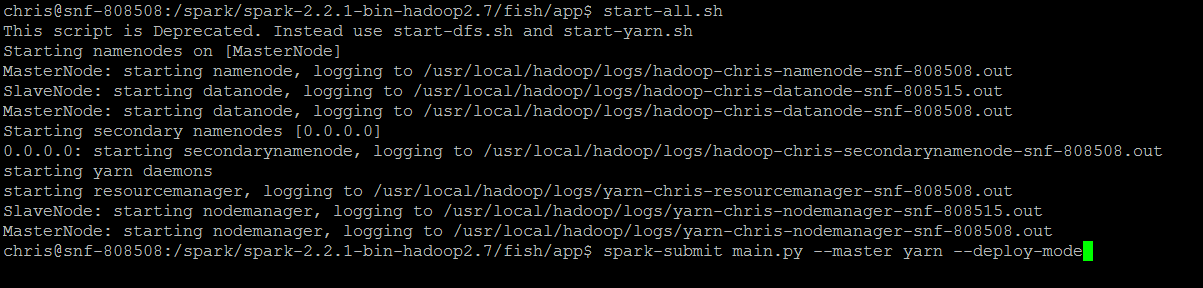
***hdfs dfs -put*** */spark/spark-2.2.1-bin-hadoop2.7/fish/app/vessels.csv /data/fish/vessels.csv*

***hdfs dfs -put*** */spark/spark-2.2.1-bin-hadoop2.7/fish/app/fish.csv /data/fish/fish.cs*v

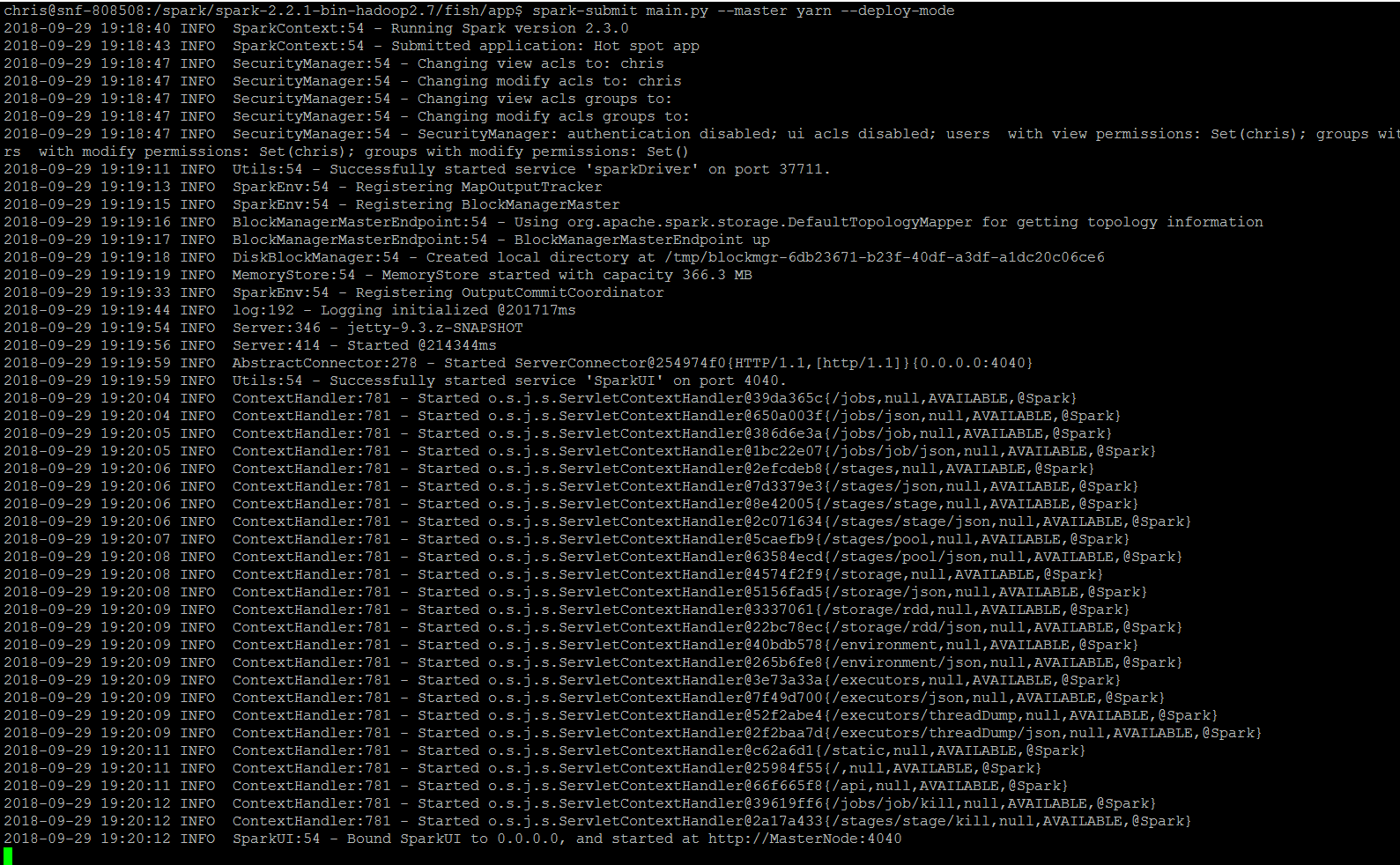
### Spark submit

Στην συνέχεια ανεβάζουμε και το python αρχείο (main.py) και το κάνουμε submit το task για να τρέξει.

*spark-submit main.py --master yarn --deploy-mode*

**

Εικόνα 1



Εικόνα 2

### 

### Το πρόγραμμα

Αρχικά τα παραπάνω paths, για τα αρχεία δεδομένων καθώς και το path για να σώσει τα αποτελέσματα, μπορούν πολύ εύκολα να αλλάξουν από τις μεθόδους *csv\_file\_path, fishing\_vessels\_csv\_file\_path, result\_path* ανάλογα.

Σε πρώτη φάση το πρόγραμμα φορτώνει τα δεδομένα των δύο αρχείων σε δύο RDD.

Στην συνέχεια γίνεται ένα left outer join ανάμεσα στο RDD με τις τροχιές των πλοίων και το RDD με τα πλοία τα οποία είναι αλιευτικά. Αυτό έχει σαν αποτέλεσμα να δημιουργείται ένα καινούριο RDD το οποίο δεν έχει κάνει αντιστοιχία στις τροχιές των πλοίων που δεν είναι αλιευτικά. Στην συνέχεια φιλτράρουμε το RDD αυτό και δημιουργούμε ένα νέο το οποίο έχει τροχιές από αλιευτικά πλοία μονο. Το νέο RDD είναι περίπου το 60% του αρχικού RDD που περιέχει όλες τις τροχιές. Την παραπάνω διαδικασία την εκτελεί η μέθοδος load\_rdd.

Αφού πλέον έχουμε τα σωστά δεδομένα το πρόγραμμα συνεχίζει με την δημιουργία του πλέγματος τριών διαστάσεων, latitude-longitude-time.

Για τις διαστάσεις των παραλληλεπιπεδων που αποτελούν το πλέγμα επιλέχθηκαν τα

0.003, 0.003, 120 για τις διαστάσεις latitude-longitude-time αντίστοιχα. Το 0.003 είναι μοίρες οπότε αντιστοιχεί σε περίπου 400 μέτρα ενώ το 120 είναι λεπτά. Οπότε το κάθε κουτί του πλέγματος είναι μια περιοχή 1,6 τετραγωνικού χιλιομέτρου για δύο ώρες, ποσά τα οποία θεωρηθηκαν αξιόλογα την συνύπαρξη αλιευτικών σκαφών σε συγκεκριμένο χώρο και χρόνο.

Είναι προφανές ότι όσο μικρότερα είναι τα παραλληλεπίπεδα τόσο πιο ακριβής είναι ο αλγόριθμος, όμως αυτό έχει ως αποτέλεσμα να επηρεάζει και την απόδοση του, όμως για πολύ μικρά ποσά δεν έχει νόημα να γίνεται ανάλυση.

Η διαδικασία συνεχίζεται με τον υπολογισμό της ελάχιστης τιμής του  χρόνου στα δεδομένα μας προκειμένου να την χρησιμοποιήσουμε στους υπολογισμούς μας στην συνέχεια. Η μεταβλητή αυτή είναι μια broadcast μεταβλητή, την οποία δημιουργήσαμε με σκοπό  να την μοιράσουμε ανάμεσα στις διεργασίες του Spark.

Επίσης για να μετασχηματίσουμε τα δεδομένα μας δημιουργούμε ένα νέο RDD (structured\_weighted\_data) το οποίο αποθηκεύεται προσωρινά στον δίσκο και στην μνήμη για γρήγορη προσπέλαση παρακάτω. Η προσωρινή αποθήκευση γίνεται με την χρήση της persist() σε mode ***StorageLevel.MEMORY\_AND\_DISK.*** Σε αυτό το βήμα δημιουργούμε πλειάδες (tuples) της μορφής time, lat, lon, xi, id, όπου το xi είναι η ευκλείδεια απόσταση του εκάστοτε σημείου από το σημείο 0,0,0. Αυτό είναι μια παραδοχή που έγινε προκειμένου να αρχικοποιηθεί η τιμή xi.

Στο επόμενο στάδιο πρέπει να υπολογίσουμε το n, δηλαδή τον αριθμό των κελιών που θα έχει το σύστημα. Οπότε υπολογίζονται οι ελάχιστες και μέγιστες τιμές καθώς και το διάστημα που κυμαίνονται τα τρία χαρακτηριστικά. Έχοντας αυτά μπορούμε να υπολογίσουμε και το n απλά πολλαπλασιάζοντας το εύρος των διαστάσεων μεταξύ τους.

Έχοντας πλέον  το n, μπορούμε να υπολογίσουμε το πλήθος των σημείων σε κάθε παραλληλεπίπεδο αλλά και το σύνολο του xi των σημείων του κάθε κελιού.

Εν ακολουθία υπολογίζεται το άθροισμα των xi και των xi^2. Αυτό γίνεται με την χρήση της foreach και των accumulators του Spark. Σε αυτό το σημείο προχωρήσαμε στη χρήση accumulator προκειμένου να μοιράσουμε τον όγκο εργασίας σε πολλά tasks. Με τον ίδιο τρόπο υπολογίσαμε και τον αριθμό των κελιών. Η παραπάνω υπολογισμοί γίνονται στις γραμμές 176-185. Οι παραπάνω τιμές χρειάζονται για να υπολογιστούν οι τιμές του X και του S.

Η διαδικασία κλιμακώνεται με τον αλγόριθμο να μας πληροφορεί ότι στην συνέχεια πρέπει να υπολογιστεί το σύνολο του xi των γειτονικών κελιών για το κάθε κελί ξεχωριστά. Μετέπειτα γίνεται ο υπολογισμός της τιμής της στατιστικής συνάρτησης Gi για κάθε κελί και με γνώμονα αυτό επιλέγονται τα κελιά με την μεγαλύτερη τιμή. Το πλήθος των κελιών που τελικά επιλέγονται είναι το 5% με την μεγαλύτερη τιμή Gi.

### Οπτικοποίηση Δεδομένων

Αφου το πρόγραμμα τελειώσει βγάζει έναν html αρχείο σαν output στο driver node.

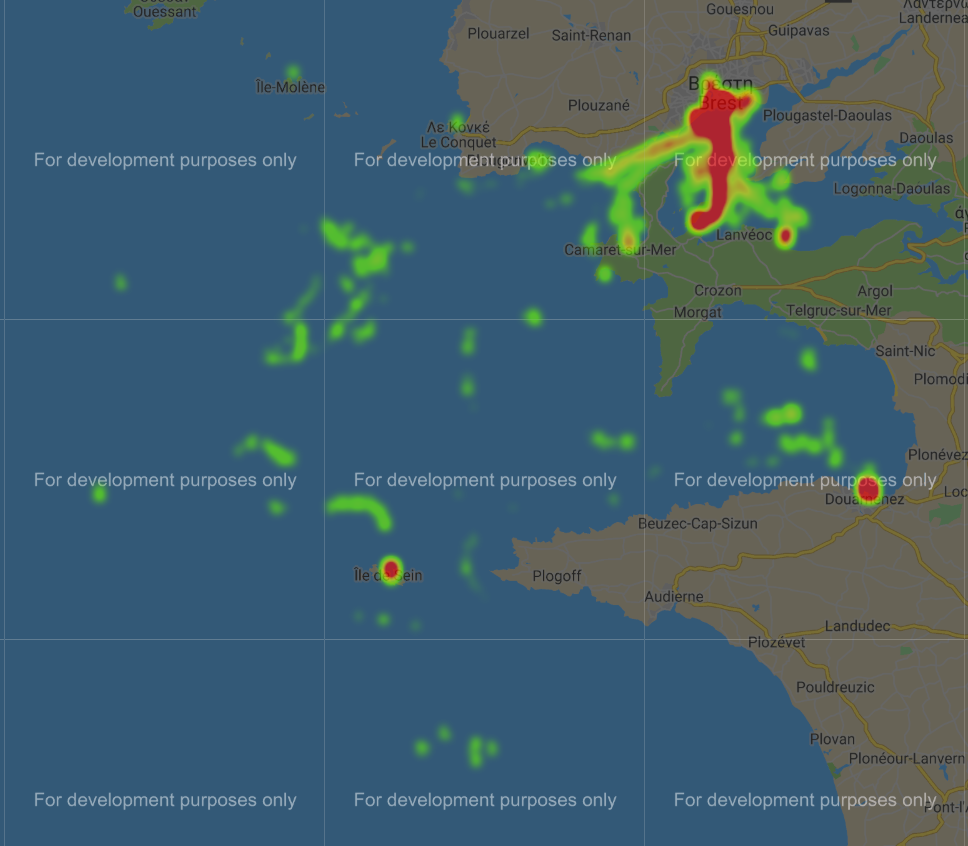
Το αρχείο αυτό εμφανίζει έναν διαδραστικό χάρτη με όλα τα σημεία ενδιαφέροντος οπότε ο χρήστης μπορεί να αναλύσει και να εξάγει συμπεράσματα.

### Χρόνος εκτέλεσης

Το πρόγραμμα αυτό εκτελείται 15 λεπτά σε single node standalone spark με 16 GB RAM, 8 cores.

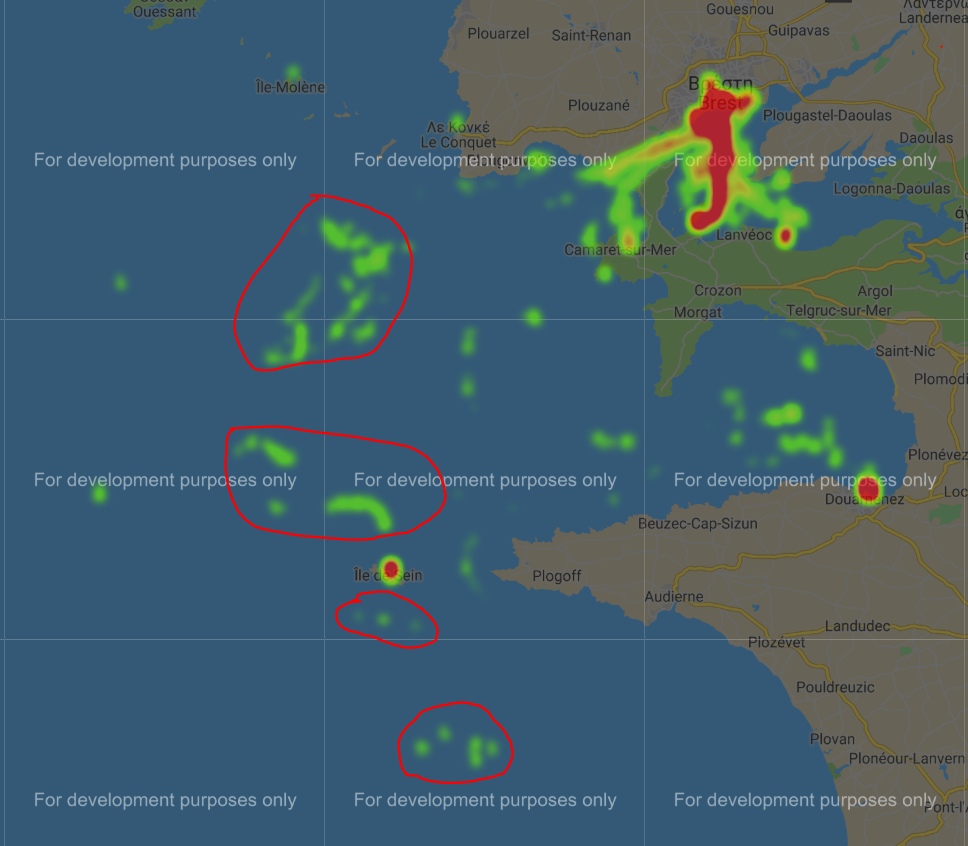
## Αποτελέσματα

Όπως φαίνεται και στους χάρτες που δημιουργεί το πρόγραμμα εμφανίζεται μεγάλη πυκνότητα σε περιοχές λιμανιών καθώς και στις διαδρομές για να βγουν στην ανοιχτή θάλασσα. Παρακάτω θα βρείτε έναν χάρτη (Εικόνα 3) που έχει προκύψει από την ανάλυση των δεδομένων που είχαμε διαθέσιμα. Οι περιοχές που κοντά στα λιμάνια αλλά και στους κόλπους κοντά σε αυτά που οδηγούν στην ανοιχτή θάλασσα είναι χρωματισμένες με κόκκινο μιας και εκει υπάρχει πολύ μεγάλη συχνότητα από αλιευτικά πλοία.



Εικόνα 3

Εκτος από τα παραπάνω σημεία εμφανίζονται και πολλά άλλα σημεία εντός της θάλασσας τα οποία ενδέχεται να είναι περιοχές που προτιμούνται για αλιεία. Στο παρακάτω χάρτη (Εικόνα 4) επισημαίνονται περιοχές που είναι στην ανοιχτή θάλασσα ή ανάμεσα σε νησιά και ενδέχεται να είναι περιοχές με ενδιαφέρον αλιευτικά.



Εικόνα 4

## Βιβλιογραφία

[1] BigCAB: Distributed Hot Spot Analysis over Big Spatio-temporal Data using Apache Spark, Nikitopoulos P., Paraskevopoulos A., Doulkeridis C., Pelekis N., Theodoridis Y., <https://www.ds.unipi.gr/prof/cdoulk/papers/gis16.pdf>

[2] Spatio-Temporal Hotspot Computation on Apache Spark, Paras Mehta, Christian Windolf, Agnès Voisard <http://parasm.com/pdf/giscup16_final.pdf>

[3] Simplification and Refinement for Speedy Spatio-temporal Hot Spot Detection Using Spark, Shangfu Peng, Hong Wei, Hao Li, Hanan Samet, <https://www.semanticscholar.org/paper/Simplification-and-Refinement-for-Speedy-Hot-Spot->(-Peng-Wei/4f806087d717cf4041b834bcc5e8f13c9e5919a4

[4] An efficient map-reduce algorithm for spatio-temporal analysis using Spark, Salles Viana Gomes Magalhães, W. Randolph Franklin, Wenli Li, Marcus Vinicius Alvim Andrade, <https://pdfs.semanticscholar.org/6e91/1a6b70c115d0f3405c63f8f61b78ce9da2e2.pdf>