# ΜΕΛΕΤΗ ΓΙΑ SCALABLE DISTANCE JOINS ΓΙΑ ΧΩΡΙΚΑ ΔΕΔΟΜΕΝΑ:

ΥΛΟΠΟΙΗΣΗ ΜΕ ΧΡΗΣΗ ΤΗΣ ΓΛΩΣΣΑΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ JAVA, ΤΟΥ FRAMEWORK SPRING-BOOT ΚΑΙ ΤΗΣ ΜΕΘΟΔΟΥ SPARK.

**Αντρέας Πατσιμάς**

*Dept. of Digital Systems University of Piraeus, Piraeus, Greece* [andreas-patsim@hotmail.com](mailto:kouvaris.t.g@gmail.com)

me2033

## Περίληψη

Η σημερινή εποχή χαρακτηρίζεται από την ταχεία ανάπτυξη της τεχνολογίας και ιδιαίτερα του Διαδικτύου. Η εξάπλωση των σύγχρονων κινητών συσκευών που κάνουν χρήση του Διαδικτύου και είναι εξοπλισμένες με GPS, ραντάρ κλπ είναι παράγοντες που καθιστούν απαραίτητη την ανάπτυξη τεχνικών επεξεργασίας μεγάλων δεδομένων και πιο ειδικά χωρικών δεδομένων.

Τεχνικές που θα επιλύουν προβλήματα όπως για παράδειγμα, ένας χρήστης να θέλει να βρει σημεία ενδιαφέροντος όπως ξενοδοχεία, εστιατόρια κλπ να δίνει μια θέση, μια ακτίνα αναζήτησης και να λαμβάνει συνδυασμούς ξενοδοχείων-εστιατορίων που να βρίσκονται εντός αυτής της ακτίνας αναζήτησης.

Ή ακόμα για πιο σύνθετα προβλήματα, όπως οι σχετικές θέσεις μεταξύ πλοίων σε συγκεκριμένα χρονικά διαστήματα εντός μιας ακτίνας αναζήτησης, όπου ο χρήστης θα λαμβάνει τις συντεταγμένες των πλοίων των οποίων οι αποστάσεις θα είναι εντός της ακτίνας αναζήτησης.

Η παραπάνω λειτουργία για σύνολα δεδομένων που έχουν μεγάλους όγκους και η επεξεργασία τους γίνεται με κεντρικό υπολογισμό είναι κάτι πολύ δύσκολα εφικτό ή ακόμη και πρακτικό. Ως εκ τούτου, η αναγκαιότητα κλιμακούμενης επεξεργασίας μεγάλων όγκων των συνόλων δεδομένων, καθιστά απαραίτητη τη χρήση τεχνικών, προκειμένου να μπορεί να παραλληλιστεί ο υπολογισμός και η επεξεργασία τους ώστε να επιτευχθεί η επεκτασιμότητα(scalability)**.**

## 2. Εισαγωγή

Η παρούσα εργασία πραγματοποιήθηκε στα πλαίσια της απαλλακτικής εργασίας του Β’ Εξαμήνου στο μάθημα «Επεξεργασία Μεγάλων Δεδομένων Τεχνικές και Εργαλεία» με θέμα: «Scalable Distance Joins for Point Data». Η εργασία στοχεύει στην ανάπτυξη συστήματος σε Apache Spark για την εκτέλεση distance join queries με αποδοτικό τρόπο που θα συμβάλει στην επίλυση προβλημάτων όπως αυτά που περιγράφονται στην Περίληψη παραπάνω.

Πάνω στο σύστημα έχει χτιστεί εφαρμογή, συγκεκριμένα ένα API, που δέχεται ως input δύο σύνολα δεδομένων σημείων Α και Β, που περιγράφονται από στήλες: (ID, x, y). Η εφαρμογή υπολογίζει τα ζεύγη (α,β), όπου α ∊ Α και β ∊ Β, για τα οποία ισχύει: d(α,β) ≤ θ, όπου θ παράμετρος που δίνεται από το χρήστη.

Επιπλέον εκτελείται για παράλληλη υλοποίηση που λειτουργεί για τεράστια σύνολα αυθαίρετων συνόλων δεδομένων και ελαχιστοποιεί το χρόνο εκτέλεσης.

Η παραπάνω εφαρμογή χτίστηκε σε γλώσσα προγραμματισμού Java και για την επίτευξη της παράλληλης υλοποίησης δημιουργήθηκε ένας cluster με δύο nodes σε virtual machines που παρέχεται στο https://okeanos.grnet.gr/ ακολουθώντας σχετικές οδηγίες από εργασία που είχε πραγματοποιηθεί από φοιτητές προηγούμενων ετών στο συγκεκριμένο μεταπτυχιακό πρόγραμμα[1].

**3. Μεθοδολογία**

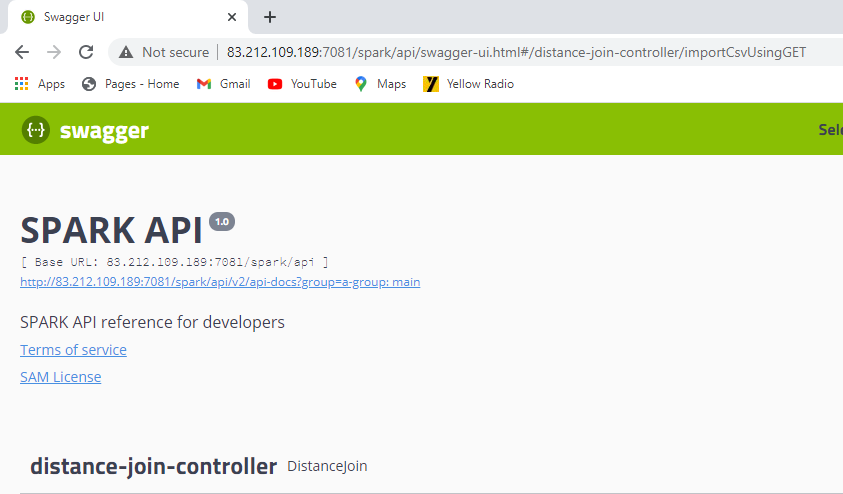
**3.1 Apache Spark - Java**

Όπως αναφέρθηκε παραπάνω για το χτίσιμο της εφαρμογής αξιοποιήθηκε η γλώσσα προγραμματισμού Java. Η εφαρμογή χτίστηκε με τη βοήθεια του IDE Intellij[2] και επιπλέον έγινε χρήση του Maven[3]. Στο πλαίσιο αυτό χρησιμοποιήθηκε το framework Spring-Boot, ενώ ταυτόχρονα έγινε εισαγωγή των libraries spark-core\_2.11 και spark-sql\_2.11 για την εγκατάσταση του Spark. Με αυτόν τον τρόπο αξιοποιήθηκαν οι δυνατότητες που προσφέρουν τα libraries αυτά για την αποτελεσματική υλοποίηση και εκτέλεση του αλγορίθμου.

Αρχικά, η εφαρμογή διαβάζει τα csv αρχεία και με τη βοήθεια του Spark μορφοποιεί τα δεδομένα σε spark dataframes. Με τη χρήση των dataframes, στη συνέχεια γίνεται καταμερισμός των δεδομένων σε slices αξιοποιώντας τις δυνατότητες που προσφέρει το spark-sql, εκτελώντας αντίστοιχο query. Ο αλγόριθμος, στο τελικό του στάδιο ξανά με την εκτέλεση query με τη βοήθεια του spark-sql συγκρίνει τα σημεία μεταξύ των δύο συνόλων δεδομένων ώστε η απόστασή τους να είναι μικρότερη της τιμής θ που δίνει ως είσοδο ο χρήστης στην εφαρμογή.

Η διαδικασία αυτή εκτελείται για κάθε slice χωριστά αλλά ταυτόχρονα σε κάθε node του cluster όπου για αυτό θα γίνει πιο αναλυτική αναφορά παρακάτω. Με αυτόν τον τρόπο επιτυγχάνεται η παραλληλία γεγονός που βελτιστοποιεί σε μεγάλο βαθμό την απόδοση και το χρόνο εκτέλεσης του αλγορίθμου όπως θα δούμε και στην ανάλυση των αποτελεσμάτων παρακάτω.

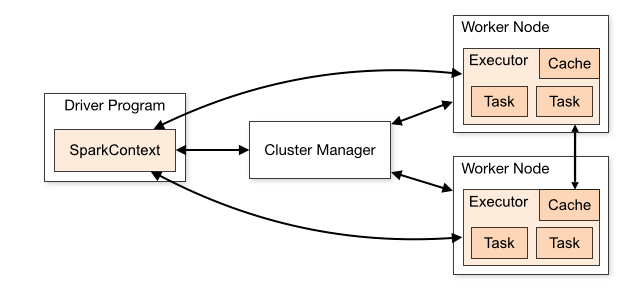
Για την καλύτερη δυνατή εκτέλεση της εφαρμογής χρησιμοποιήθηκε το library springfox-swagger-ui. Το library αυτό είναι ένα User Interface μέσω του οποίου μπορεί ένας χρήστης να δουλεύει πάνω στο API της εφαρμογής. Ουσιαστικά λειτουργεί σαν client μέσω του οποίου ο χρήστης μπορεί να κάνει κλήσεις στο server και να δέχεται τα αντίστοιχα αποτελέσματα που προσφέρει η εφαμογή. Στην προκειμένη περίπτωση ο χρήστης δέχεται τα αποτελέσματα που προκύπτουν από την εκτέλεση του αλγορίθμου για τα distance joins. Παρακάτω παρατίθεται ενδεικτικά μια εικόνα της εφαρμογής.



**3.2 Apache Spark - Cluster**

Οι εφαρμογές Spark που εκτελούνται ως ανεξάρτητα σύνολα διαδικασιών σε έναν cluster, συντονίζονται από το αντικείμενο SparkContext στο κύριο πρόγραμμά.

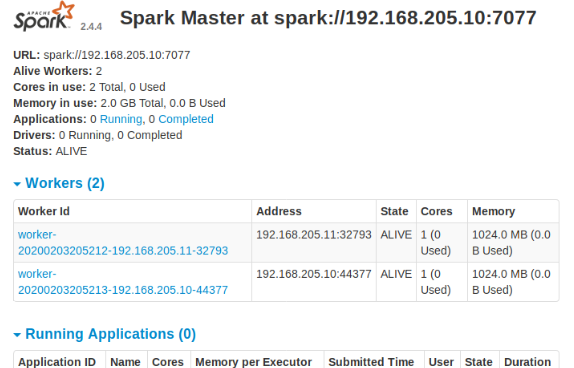
Συγκεκριμένα, για να τρέξει σε έναν cluster, το SparkContext μπορεί να συνδεθεί είτε αυτόνομα(standalone) είτε μέσω του Yarn όπου κατά αυτόν τον τρόπο κατανέμονται οι πόροι σε όλες τις εφαρμογές. Μόλις γίνει η σύνδεση, το Spark δίνει τη δυνατότητα στα nodes που απαρτίζουν τον cluster να τρέξουν διαδικασίες που εκτελούν υπολογισμούς και αποθηκεύουν δεδομένα για την εφαρμογή. Στη συνέχεια, στέλνει τον κώδικα της εφαρμογής στα nodes ο οποίος ορίζεται από αρχείο JAR το οποίο έχει μεταφερθεί στο SparkContex. Τέλος, το SparkContext στέλνει εργασίες στα nodes για εκτέλεση.



Για την πραγματοποίηση της έρευνάς χρησιμοποιήθηκε ως Infastructure το okeanos.grnet.gr. Εκεί δημιουργήθηκαν 3 συνολικά Virtual Machines (VMs).

Το 1ο μηχάνημα χρησιμοποιήθηκε για την φιλοξενία του Spring Boot Rest API, το οποίο επικοινωνεί με το context του Spark, το οποίο έγινε εγκατάσταση με τη βοήθεια σχετικών οδηγιών[6].

Το λειτουργικό του σύστημα είναι Ubuntu Desktop, έχει 4 CPU’s, 8GB RAM και 5GB disκ space. Τα υπόλοιπα 2 μηχανήματα υποστηρίζουν τα αντίστοιχα spark nodes. Όλα τα VMs τρέχουν λειτουργικό σύστημα Ubuntu 16.04 και έχουν 4 CPU’s. 8GB RAM και 5GB δίσκο. Τα VM’s ανήκουν όλα στο ίδιο δίκτυο εκτός από το 1ο το οποίο βρίσκεται σε διαφορετικό για λόγους ασφαλείας. Επίσης αξίζει να σημειωθεί πως το 1ο παίζει το ρόλο του master και είναι σε public ip ενώ τα άλλα 2 σε private.

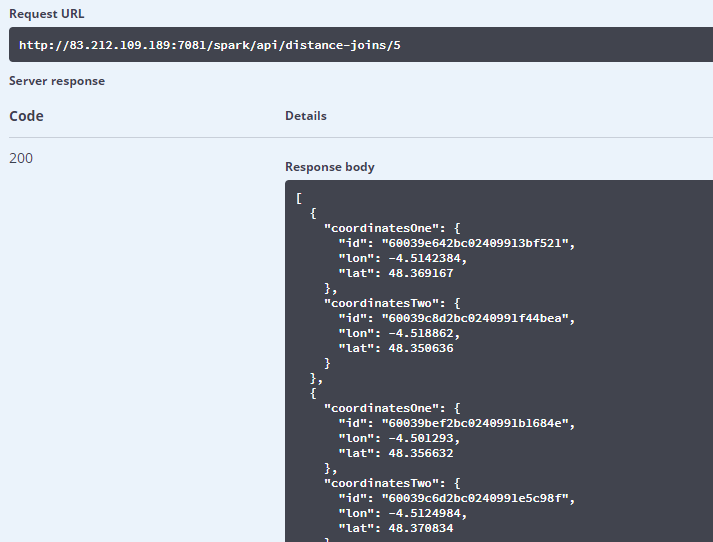


**4. Αποτελέσματα**

Τα δεδομένα, πάνω στα οποία εξετάστηκε η λειτουργικότητα της εφαρμογής, αποτελούνται από χώρο-χρονικά σημεία πλοίων που καλύπτουν μια διάρκεια 6 μηνών από 1/10/2015 έως 31/3/2016 στη θαλάσσια περιοχή ανοιχτά του Μπρέστ της Γαλλίας. Περιλαμβάνονται επίσης και δεδομένα πλοήγησης καθώς και τα σταθερά δεδομένα κάθε πλοίου (όνομα, σημαία, διεθνής κωδικός πλοίου κ.ο.κ.) Η συλλογή των δεδομένων έγινε μέσω του συστήματος ΑΙS (Automatic Identification System)Για την εκτέλεση του αλγορίθμου αξιοποιήθηκαν δύο μεγάλα σύνολα δεδομένων. Με την κατάλληλη προ-επεξεργασία των δεδομένων, προέκυψαν 2 μεγάλα σύνολα δεδομένων σε μορφή csv αρχείου με στήλες \_id, X, Y, όπου \_id είναι ο μοναδικός κωδικός του κάθε πλοίου και X,Y οι συντεταγμένες των θέσεων των πλοίων.

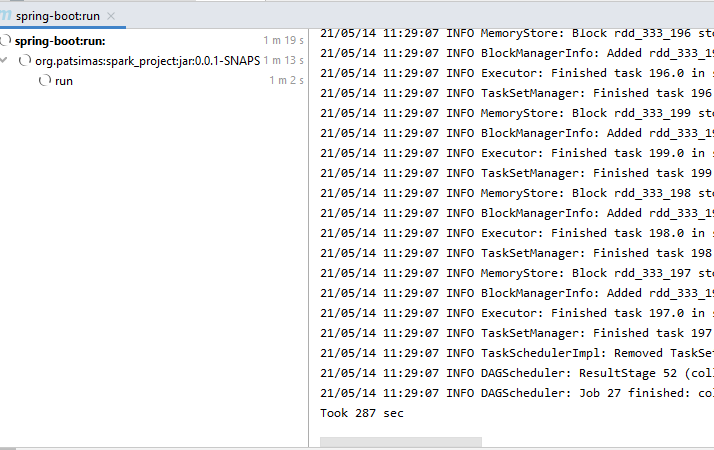
Αφού εκτελεστεί ο αλγόριθμος distance join, η εφαρμογή επιστρέφει τα ζεύγη των σημείων για κάθε πλοίο που απέχουν απόσταση μικρότερη της τιμής που καταχωρεί ο χρήστης. Τα αποτελέσματα είναι σε μορφή json αρχείου.

Παρακάτω παρατίθεται σχετική εικόνα των αποτελεσμάτων:

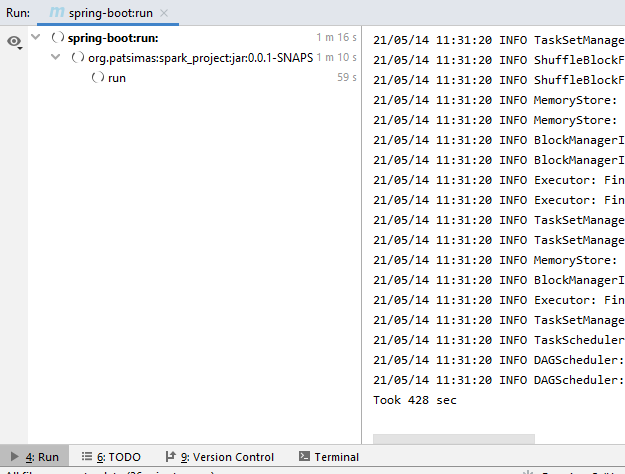


Ο αλγόριθμος αρχικά εκτελέστηκε τοπικά στον υπολογιστή και στη συνέχεια στον cluster στο Virtual Machine από το okeanos.grnet.gr. Επιλέχθηκε η εκτέλεση να γίνει και με τους δύο παραπάνω τρόπους ώστε να γίνει σύγκριση των αποτελεσμάτων. Όπως είναι λογικό η εκτέλεση στον cluster έγινε με μεγαλύτερη ταχύτητα από ότι τοπικά. Χαρακτηριστικά στον cluster η ταχύτητα εκτέλεσης ήταν στα 287 δευτερόλεπτα ενώ τοπικά στα 428 δευτερόλεπτα.

Χρόνος εκτέλεσης στον cluster:



Χρόνος εκτέλεσης τοπικά:



**5. Συμπεράσματα**

Μία σύγχρονη εφαρμογή, όπως αυτή που υλοποιήθηκε στην εργασία αυτή, με μεγάλα δεδομένα θέτει μεγάλες απαιτήσεις, με αποτέλεσμα η υποστήριξή της από ένα απλό σχεσιακό μοντέλο να είναι αρκετά δύσκολη. Σύγχρονες τεχνικές επεξεργασίας μεγάλων δεδομένων όπως αυτή του Spark, προσφέρει μεγάλες δυνατότητες όπως η επεξεργασία τους να μην προαπαιτεί κάποιο σχήμα βάσης. Ενώ ταυτόχρονα με το Spark-SQL να μπορούν να χρησιμοποιηθούν αντίστοιχα sql queries για την ανάκτηση δεδομένων στη μορφή που επιθυμεί ο χρήστης. Όπως είδαμε προσφέρει δυνατότητες όπως αυτήν της παράλληλης εκτέλεσης, πράγμα που επιτρέπει την ταχύτερη εκτέλεση μιας διαδικασίας όπως είδαμε και παραπάνω στο παράδειγμα με τα distance joins.

Μελλοντικά ως μία επέκταση της εργασίας και ένας διαφορετικός τρόπος προσέγγισης θα μπορούσε να είναι η εκτέλεση του αλγορίθμου, με τη βοήθεια του Spark, σε μη σημειακά δεδομένα λαμβάνοντας υπόψη ακόμα και κείμενο.

**6. Βιβλιογραφία**

[1] ΚΑΤΑΝΕΜΗΜΕΝΗ ΛΕΙΤΟΥΡΓΙΑ APACHE SPARK ΜΕ HADOOP DISTRIBUTED FILE SYSTEM (HDFS)

[2] <https://www.jetbrains.com/help/idea/discover-intellij-idea.html>

[3] <https://maven.apache.org/guides/>

[4] A survey of trajectory distance measures and performance evaluation

Han Su · Shuncheng Liu· Bolong Zheng · Xiaofang Zhou · Kai Zheng

[5] Efficient distance join query processing in distributed spatial data management systems

Francisco García -García, Antonio Corral, Luis Iribarne, Michael Vassilakopoulos, Yannis Manolopoulos

<https://www.sciencedirect.com/science/article/abs/pii/S0020025519309855>

[6] https://medium.com/@jootorres\_11979/how-to-install-and-set-up-an-apache-spark-cluster-on-hadoop-18-04-b4d70650ed42