# 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 μια 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’ Εξαμήνουστο 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 machine 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[4]. 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.

Δσσσσσσδσδνδξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξξdddcddddddddddddddddddddddddddddddddddddddddd

**3.2 Apache Spark - Cluster**

Dddddddddddddddddddddddddddddddddddddddddddweeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeewccd

**4. Αποτελέσματα**

Lllmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmvσδδδδδδδδδδδδδδδδδδδδδδδδδδδδδδδδδδδδ

**5. Συμπεράσματα**

Σσσσσσσσσσςςςςςςςςςςςςςςςςςςςεεεεεεεεεεεεεεεεεεεεεεεεεεεεεεεεεεεεεεεεε

**6. Βιβλιογραφία**

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